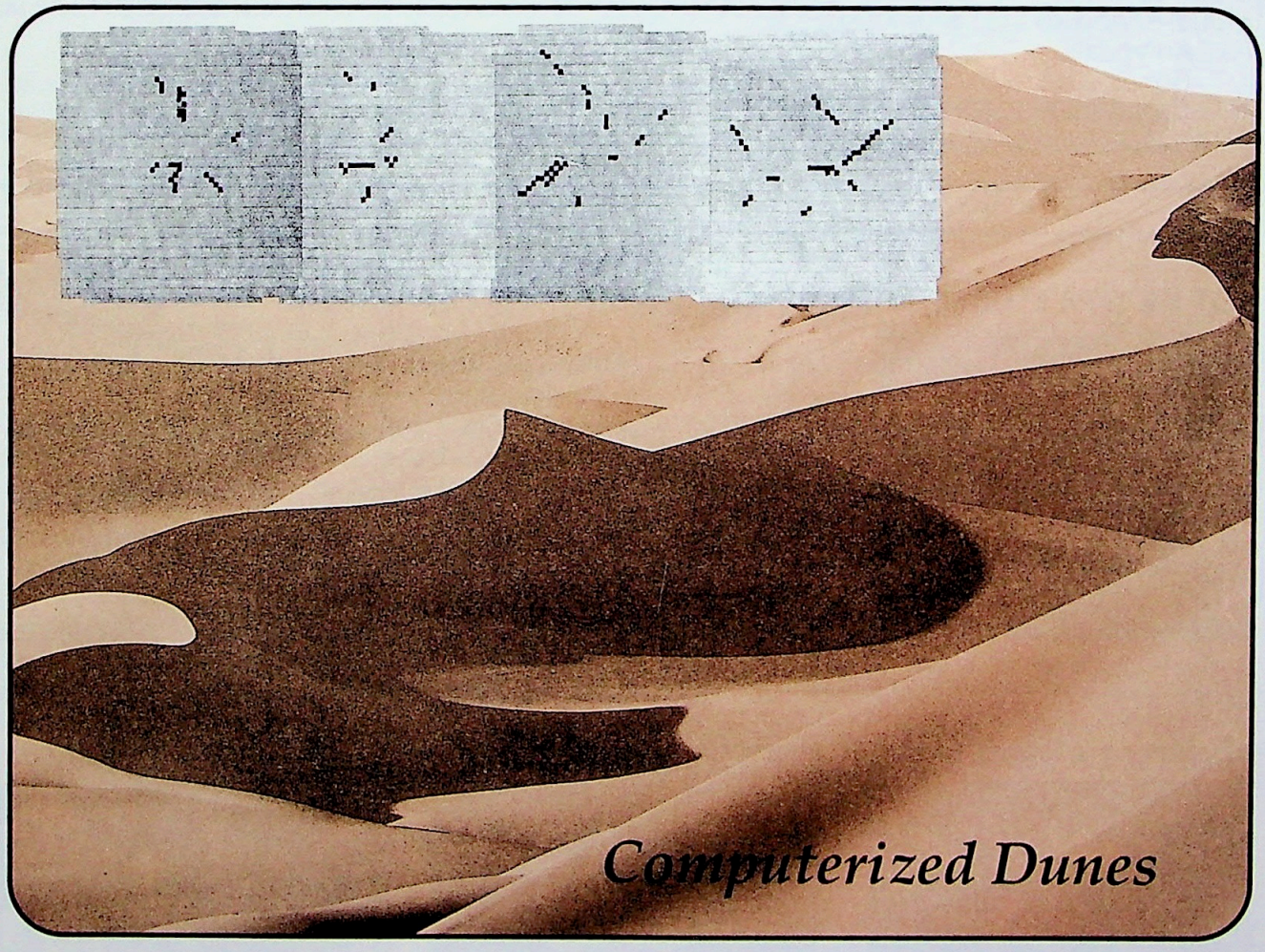
JULY/AUGUST1990

Algorithm

1.5

THE PERSONAL PROGRAMMING NEWSLETTER



Computerized Dunes

Mathemagical Black Holes
Type Fonts that Grow
Switching Packets and Trains
The Adventures of Vat Man

When you get your first issue of Amygdala you'll wonder how you ever did without it!

What's "Amygdala"?

Amygdala's a newsletter. It's devoted to fractals like the Mandelbrot set & Julia sets as well as other iteration sets.

What will I find in it?

- Fractal pictures;
- Articles about generating fractals on computers;
- Fractal bibliography;
- Reviews of fractal programs;
- Elementary & advanced articles about the math background underlying fractals.
- Et cetera...

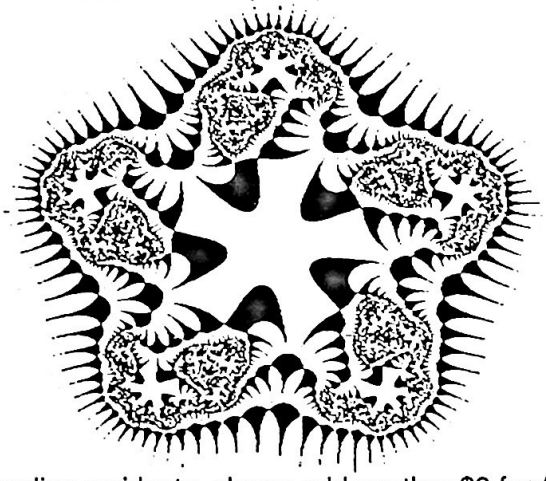
How can I get it?

- Just send $25 US for ten issues to:

  Amygdala
  Box 219
  San Cristobal, NM 87564 (USA)

Check, credit card, money order, cash...

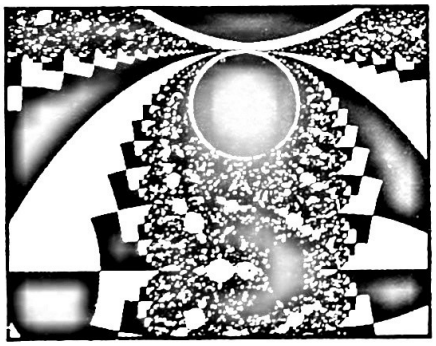
A fractal biomorph by Ian Entwistle



Canadian residents, please add another $3 for Air Mail postage.

An additional $32 will get you the color slide supplement–forty incredible fractal slides. Each set of four 35 mm slides accompanies an issue of the newsletter.

LOVER'S PATHS



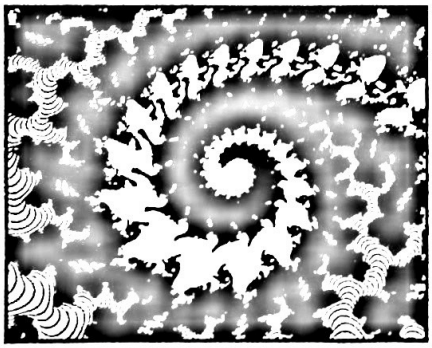
COLOR T-SHIRT $12.00US

Small, Medium, Large, X-Large.
100% white heavy cotton, brilliant color, high res.
Please add $3.00US P&H to your entire shirt order.
Mastercard, Visa, Check, MO, PO. 100% Guaranteed.
If you want to sell them, ask for wholesale information.
Black and white images are only approximations.

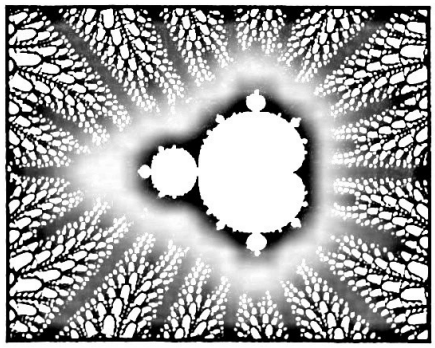
'MANDELBROT SETS and JULIA SETS'
2 hour video, $50.00US+ $3.00US P&H VHS, Beta

ART MATRIX PO 880AL Ithaca, NY 14851-0880 USA
1 800 PAX-DUTY or (607) 277-0959
'The Paths of Lovers Cross in the Line of Duty.'

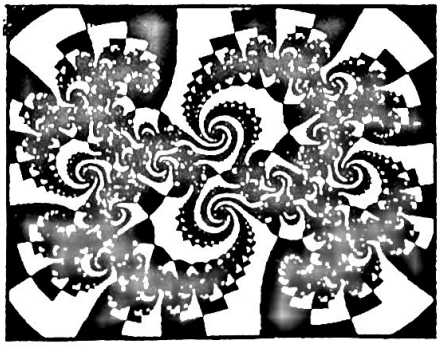
NAUTILUS SHELL



MANDELBROT SET



ESCHER'S JULIA



Algorithm

THE PERSONAL PROGRAMMING NEWSLETTER

Algostaff

Editor
A.K. Dewdney

Assistant Editors
Jennifer Connor
John O'Hagan

Managing Editor
Louis D. Magguilli

Layout & Design
Tammy Golinski

Production Services
Bogdan Graphics Inc.

ALGOLETTERS - The Readers
Fractal Vegetables
Distributed Problems
Algorithmic Issues
Creeping Commodores
Average Roundup ____ 2

EASY PIECES - Michael W. Ecker
Wherein numbers fall into mathemagical black holes ____ 4

COMPUTER ADVENTURES - A.K. Dewdney
Critical events in a computer sandpile ____ 7

PERSONAL PROGRAMS - Clifford A. Pickover
Growing your own fonts ____ 11

ALGOPUZZLE - Dennis Shasha
One man's poison ____ 13

ALGOFACT - Michael W. Lesk
Packets versus circuits: a 19th century case study ____ 16

ALGOFICTION - A.K. Dewdney
Vat Man (and Reuben the Boy Wunder) ____ 20

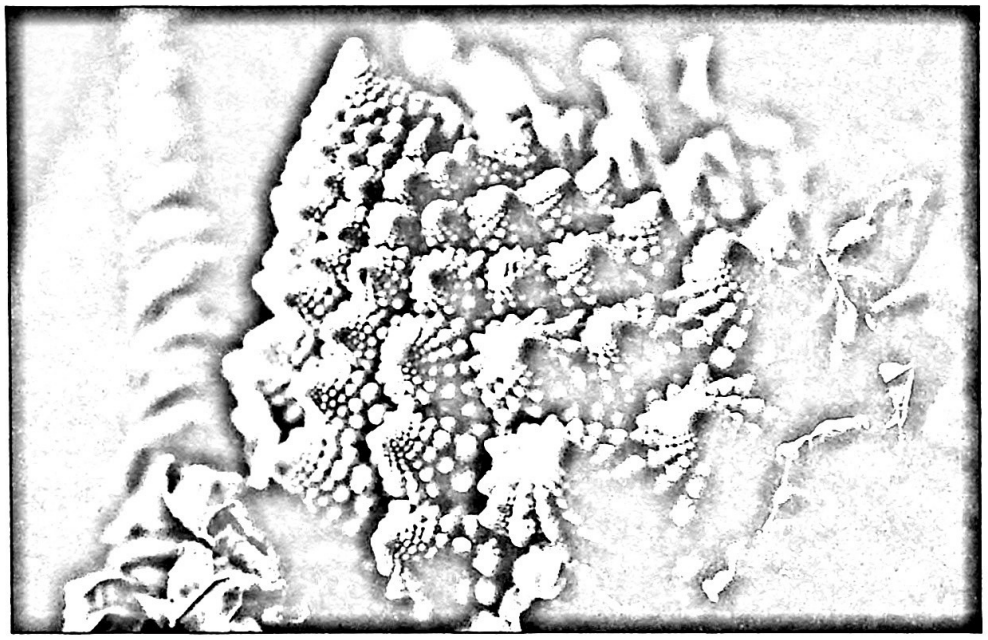
MICRO SCOPE - David Geoffrey
The Railrodders
A Gallery of Video Zooms ____ 25

BULLETIN BOARD
Software, games and publications from individuals and companies ____ 28

Address all editorial correspondence to A.K. Dewdney, 2 Oxford Street West, London, Ontario N6H 1P9, CANADA. Address advertising, business and all other correspondence to Louis D. Magguilli, Algorithm, P.O. Box 29237, Westmount Postal Outlet, 785 Wonderland Road S., London, Ontario N6K 1M6. Owned and published bimonthly by 854832 Ontario Inc., doing business as *Algorithm*. All rights reserved, including the right to accept or reject all editorial and advertising matter. Unsolicited materials must be accompanied by self-addressed return envelopes and postage or dollar equivalent. No part of this newsletter may be reproduced without the written permission of *Algorithm*. Copyright© 1990 by *Algorithm*. Mail subscriptions: $29.95 (U.S.) per year; single copies $5.00 (U.S.). Overseas add $4.00 (U.S.) for shipping. ISSN 0843-9753. Algophone (519) 432-8042. Mailed July 1990. Second Class Mail Permit Number 8561. Postage paid in London, Ontario.

ALGOLETTERS

by The Readers



FRACTAL VEGETABLES

Here is a picture of a somewhat weird plant called "Romanesco Cauliflower." The picture was taken by Marjorie Ireland, a graduate student here in biostatistics, who found this specimen in the produce section of a local fancy gourmet supermarket. When Marjorie first brought in the picture she said "look - it's a fractal plant!" We examined the plant itself with a fairly powerful magnifier. Each little sub-bump that we could resolve repeated the pattern of the whole thing. In this sense it really does appear to be fractal, unlike, say, a sunflower head. The plant must have something quite peculiar in its DNA. Thinking of DNA as essentially a computer program, does Romanesco Cauliflower DNA contain some kind of infinite loop or recursive subroutine?

Coincidentally I noticed the plant in a garden seed catalog (Park Seed Co., Cokesbury Road, Greenwood, S.C. 29647; $1.35 for a seed packet. They claim it is favored by Italian chefs.) So even people without a computer can grow their own fractals.

It tastes, I would say, not quite as good as regular cauliflower but better than broccoli.

John E. Connett, Ph. D.
Coordinating Centers for
Biometric Research,
University of Minnesota,
Minneapolis, MN

DISTRIBUTED PROBLEMS

There exists a class of problems which are good candidates for discussion and solution by the readers of this magazine. These problems may be useful, interesting, playful or superfluous. However they are viewed, I believe we have the talent and resources available to make a major contribution to the art of computation and to society in general.

What are these problems? Programs which require considerable computer time to solve may require innovative programming techniques.

Why are they not being solved? It takes time and money to run computers. Business wants its computers used for work. Education does not have the resources. The military does not want the intrusion.

What is left? Thousands of home computer users with their thousands of home computers. Nearly all of these spend most of their time idling while the user slowly types away or the printer slowly prints away.

I propose the following, that:

A national clearing house be established to coordinate problems and problem solvers.

A background program framework be developed and published. This framework could then be expanded to solve a particular problem.

The computer magazines endorse and publish this proposal and that progress reports be published twice per year.

Your responses are welcome. *Algorithm* will forward them to me.

Robert Arthur
Fremont, California

ALGORITHMIC ISSUES

I do not see much in the way of discussion of algorithms in your first few issues, either in a general or theoretical sense. Perhaps some of my suggestions below will clarify what I mean.

I think a sidebar explaining your pseudo-code syntax and symbolism should be a regular feature. Such things as conventions for separating two or more commands on the same line should be clearly stated. Likewise, whatever convention you use to indicate blocks of code, be it "begin /end," "loop / endloop," by indentation alone, or whatever, should be spelled out. (I personally find it helpful to have some flag alert me to the start of a block.) I think you are in a position to exercise some leadership here, to set a standard. I suppose symbols for mathematical operations could be stated explicitly, too. Is it silly for me to assume that some readers may not know that '*' is a programming convention for

multiplication? The assignment operator needs to be explained, so that expressions like "A <-- B-C" make some sense. Anyway, I think a regular sidebar nailing all this stuff down is a good idea.

In *Algorithm* 1.1 I was able to follow all of the algorithms, but it was at times difficult because the amount of indentation delimiting blocks was inconsistent. For example, on page 11 the Mandelbrot algorithm appeared to have more indentation at line 8 due to typographical variation, than at line 5, where it was intentional. The situation is much improved in *Algorithm* 1.3.

Here are some other questions and ideas; they might even warrant articles:

What about internal documentation or comments within an algorithm? What's your preference for your own programs? For algorithms published in the magazine?

What about the dreaded GOTO statement? Is it a legitimate step in an algorithm? Is "exit loop" a euphemism for this?

How about an article discussing alternate algorithms for the same effect, like how to write a WHILE loop without the WHILE, or REPEAT - UNTIL, for the beginners?

Now this one is important to me: What about meaningful names for variables? All of the algorithms I've seen in the magazine so far use standard algebraic notation, lots of 'A' and 'B' and 'C'. Do you have a preference, an editor's policy? I think the ability of high level computer languages to handle variable names which are more than one character long has definitely offered programmer and student alike the ability to work with understandable code. "D = S * T" is much less clear than "Distance = Speed * Time."

Finally, I would like to see an article about the relationship between data structure and code. In some cases data is simple, a couple of integer or real number variables will do. In others, the storage of information required by the algorithm is more demanding, like the cards in a deck or points on a plane. And at the other extreme, data structures for things like poker-dice or the moves of pieces in a board game are quite difficult to represent. In such cases, the rules of the game inform the data structure and the algorithm equally. I would really appreciate an article by an expert prepared to elucidate the finer points of this concept.

I really look forward to each issue. Thanks for all the fun.

Stephen B. Ringle
Dover Foxcraft, Maine

We have started to follow Mr. Ringle's suggestions by including a special How to Read Algorithms box in each issue.

Editor

CREEPING COMMODORES

As a computer owner, I always read the PERSONAL PROGRAMMING section with great interest, but as someone who has a Commodore 64, I have much trouble in running the programs. In other words, the computer is just too slow. For instance, after programming the Even-Odd Growth cellular automation (*Algorithm* 1.3), it took me approximately 2 hours to reach the 35th generation. (Of course I had implemented a tested subroutine so that the program would not waste its time on irrelevant pixels, or else I would have had to wait much much longer).

Obviously, with such slowness it's impossible to discover different variations by trial and error. Isn't there any way out?

Mohiedean Ghofrani
Tehran, Iran

See the Mailbag in Personal Programs. Dr. Pickover will forward your letter and program to a helpful soul.

Editor

AVERAGE ROUNDUP

On the subject of integer division, as in computing the average state of a cellular automaton's neighborhood:

Some readers of *Algorithm* may be interested in the following discoveries that I've made in developing QuickBASIC programs for what I call Toroidal Jones and Annular Jones -- not related to Indiana Jones but closely related to keeping up with . . . :

- QuickBASIC 4.5's CINT function rounds *n.5* to the nearest even integer. (IBM BASICA 3.3's CINT function always rounds *n.5* up.)
- QuickBASIC's CINT function is slightly faster than its INT function.
- At least with compiled QuickBASIC in my IBM PS/2 286 with Intel's 80287-10 coprocessor, A\B is nearly three times faster than INT (A/B) and produces exactly the same results if A and B are integers.
- (A + B/2) \ B will always round *n.5* up and is much faster than INT((A/B) + .5).

Thus, for a CA that involves integer averages of integer states, one can have three versions that produce very different results, depending on whether *n.5* is rounded to the nearest even integer or always rounded up, or all fractions are dropped. And for *any* version, one should *never* use the INT function!

Donovan Smith
El Cerrito, CA

EASY PIECES

by Michael W. Ecker

Wherein numbers fall into mathemagical black holes

There are black holes in outer space but there are also black holes in the inner space we call mathematics. Strangely enough, if we count down for the blastoff into the latter realm, 3-2-1, we end up in the black hole 123.

For example, start with any number, say 1248135. Now, count the number of even digits (3), the number of odd digits (4), and the total number of digits (7). the resulting number, 347, is the next member of the sequence that begins with 1248135. Now, repeat the same process on 347. The numbers of even digits, odd digits and total number of digits are 1, 2 and 3, respectively. The third member of the sequence is 123 and there the sequence ends: No matter how many times we repeat the process, we always get the same number, 123.

Was this an accident? What happens if we start with a much bigger number, say

830922517490923447183426391121010947383 7

The number of even digits is 18, the number of odd digits is 22 and the total number of digits is therefore 40. The new number is 182240. The next number? 516. The next? 123.

The process just outlined seems to exert an inexorable force on all numbers, collapsing a whole universe of integers into the mathemagical black hole, 123.

Perhaps it's time to write a program to automate our exploration of this black hole. The essential structure of the program is captured by the following algorithm:

Input the number, *n*, to be tested
Initialize variables *odd* and *even*
For each digit in the number
    if the digit is even
        then add 1 to *even*
        else add 1 to *odd*
    Print the three numbers
    Assemble the new number

Once we have an algorithm in place, a BASIC program such as the one in the box on page 5 is as easy to write as 123. For the sake of brevity, I have avoided using unnecessary variables such as in line 100 where I print EVEN + ODD instead of using an extra variable called TOTAL. The first line, 10, prompts for the input of the number N on which to test what might be called the 123 black hole hypothesis. Note that I've used not N but N$, the string version of the variable. This will store the number N as a sequence of numerical characters which we can then get at using some of the BASIC string-handling commands such as MID$ in line 50. Line 20 checks whether the VALue of the number is less than 1 or is not an INTeger. If either is the case, the program goes back to line 10 for a new number. So much comprises the "input" part of the algorithm.

The variables ODD and EVEN are initialized to 0 in line 30.

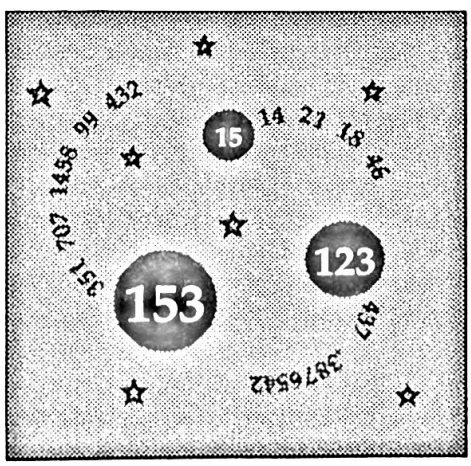
The FOR-loop in the algorithm translates into lines 40 through 70 in the program. The BASIC command LEN produces the LENgth of a string so that the variable called DIGIT can run from 1 to LEN(N$), making it possible to access the proper number of digits. Inside the FOR-loop, the command MID$ isolates digit number DIGIT (so to speak) inside the string N$. The isolated character stored in the variable D$ is then tested in line 60 to see whether it happens to be an even number. Then the program moves on to the NEXT statement, sending it back up to the beginning of the loop, all ready to process the next character. In line 60 we find the key test: Is the digit D odd or not? Since D is stored as the character D$, we use the VAL command to find the value of the digit. If the value is even then division by 2 will produce an integer and the INT function will not change the answer. Equality in the test therefore means that the number is even. Otherwise it is odd. I have separated the THEN and ELSE branches of this test to reflect the algorithm but in an actual program they should be on the same line. In the first case the program adds 1 to the count of even numbers. In the second, it adds 1 to the count of odd ones. The NEXT statement ends the FOR-loop.

Some versions of BASIC have been known to insert the occasional blank when inputting strings. To avoid the disrupting effect this might have on a program, readers are free to insert an additional command between lines 50 and 60:

55 IF D$ = " " THEN GOTO 70

Printing the three numbers, as in the algorithm, translates into lines 80, 90 and 100 where the headings EVEN, ODD, TOTAL are printed and, under them, the three current values separated by blanks.

Finally, in line 110 a NU string is put together by using + to join up the string of digits comprising the number EVEN, the string of digits in the value for ODD and the digits of their sum. If the new string equals the old one, the program ends since it is now repeating itself. If not, the new string of digits replaces the old one and the program begins executing all over again at line 30.



Black Hole Program (123)

```
10   INPUT "TYPE YOUR INTEGER"; N$
20   IF VAL(N$) < 1 OR VAL(N$) <> INT(VAL(N$)) THEN 10
30   EVEN = 0: ODD = 0
40   FOR DIGIT = 1 TO LEN(N$)
50   D$ = MID$(N$, DIGIT, 1)
60   IF VAL(D$)/2 = INT(VAL(D$)/2) THEN EVEN = EVEN + 1
                                   ELSE ODD = ODD+1
70   NEXT
80   PRINT "EVEN, ODD, TOTAL"
90   PRINT EVEN; " " ; ODD; " " ; EVEN + ODD
100  PRINT
110  NU$ = STR$(EVEN) + STR$(ODD) + STR$(EVEN + ODD)
120  IF VAL(NU$) = VAL(N$) THEN END
130  N$ = NU$
140  GOTO 30
```

In general the use of GOTOs is frowned upon because it tends to make program code very difficult to read. It may also give rise to infinite loops. Will it in this case? If it does you will have found a number that escapes the black hole.

Remember to save a copy of this black hole program. Save it, perhaps, as 123.BAS and file it via the menu system I described in *Algorithm* 1.4. That way you can retrieve the program easily and use odd moments to test new numbers.

What are Mathemagical Black Holes?

The number 123 acts as a black hole in two ways when you run the process just described:

1. Once the sequence hits 123, it never gets out, just as when a particle enters a physical black hole it never gets out.
2. Every element subject to the force of the black hole is eventually pulled into it. In this case, every number to which the iteration process can be applied will eventually end in 123.

How can I be sure that every number will fall into this particular black hole? A little mathematical argument will justify my confidence: First, notice that when the three numbers are extracted from the current number, the new number composed of the three counts is always shorter than the current one. For example, the number 1234567890 becomes the much smaller number 5510, reflecting 5 evens, 5 odds, 10 digits altogether. The shrinkage of any number with more than three digits is inevitable. Think about it for a moment and it should be at least intuitively clear. If you want to try to prove it, that is even better. You may even discover a logarithmic rule of shrinkage.

But if all numbers get pulled down to the size of three digits, we are not quite home. After all, we will only have proved that all numbers end up somewhere between 0 and 999. This leaves us with a thousand cases to check. You may be able to modify the 123 black hole program to carry out just this research goal by inserting an extra loop on the outside of the program and removing the INPUT statement.

But a little analysis will take us to the goal more quickly. First, there cannot be fewer than three digits because the iteration process always produces three numbers. Even if one of the numbers is 0, it still counts as a digit. Only one of the digits can be 0 and this leaves only four possibilities when the process reaches the three-digit stage:

033, 123, 213, 303

In other words there could be no evens and three odds, one even and two odds, two evens and one odd, or three evens and no odds. But all four of these numbers have one even, two odd numbers and a total of three digits! 123.

I define a mathemagical black hole as follows: It is a function f, a number h and a set U containing h such that:

1. f(h) = h, and
2. for each number x, in U, there is a positive integer k such that $f^k(x) = h$.

In other words, if the function is iterated k times, it will produce h, the "hole."

This definition could be made more general by changing condition 2 into a limit statement, but it captures not only the mathemagical black hole we have just been working with, but others as well.

Other Black Holes

Readers may recall the hailstone numbers that I described in *Algorithm* 1.1. There we started with a positive integer. If it was even, we divided it by 2. If it was odd, we multiplied it by 3 and added 1. Although the sequence of numbers that resulted might at first go up and down for some time, they eventually ended up in the cycle 4-2-1-4-2-1 and so on. This cycle is not a black hole but the process can be redefined so that a black hole results. Simply make an additional rule that whenever the number is even, we divide by 2 until this is no longer possible.

The number 153 turns out to be another black hole if we iterate the following procedure beginning with any number that happens to be a multiple of 3.

1. Extract each digit and cube it
2. Add the cubes to form the next number
3. Go back to step 1.

Let us take a number from a hat, one that happens to be a multiple of 3, say 432. The cubes of the digits are 64, 27, and 8. Their sum is 99. When the process is iterated, we get 1458. This in turn leads to 702, then 351 and then 153. From this point on, continued iteration produces 153 over and over again. After all, 1 + 125 + 27 = 153.

Does this always happen? Readers who are not mathematical geniuses may as well write a program along the lines of the following algorithm:

```
Input the number, num
check that num is a multiple of 3
while num <> 153
    sum <-- 0
    for each digit in num
        dcubed <-- cube of digit
        sum <-- sum + dcubed
```

Here we use the variable *num* to hold the number input by the user. as in the first program, we check that *num* is legal: If it's not a multiple of 3 the program branches back to the beginning and asks for another number. Next, the BASIC loop command WHILE controls the iteration process. Again and again, new values for *num* will be computed. WHILE (as long as) these values do not equal 153, the program will continue to add up the cubes of the digits in *num*. The left-pointing arrows, by the way, simply mean the BASIC assignment command denoted by the = sign.

One question remains: How will our program extract the successive digits of *num*? The answer is simple. If we divide *num* by 10, round down to the nearest integer and then multiply by 10 again, the resulting number will have its last digit zeroed. For example, if we apply INT to 936/10 = 93.6 we get 93. Multiplying by 10 we get 930.

Notice that

936 - 930 = 6,

the last digit. If the program saves the intermediate number 96 under the name *numb*, the formula would be

digit <-- *num* - 10* *numb*

The program would then cube the digit, add it to the ongoing sum, replace *num* by *numb* and start all over again. The loop would not really be a FOR-loop. In fact, it would also be a while loop: WHILE NUMB > 0. Don't forget, by the way, that each WHILE loop must end with WEND. Will this be enough advice for some readers to complete the program? Please let me know how your experiment goes. I would be especially surprised if you ever find a number that does not fall into the mysterious 153 black hole!

I will close with one more mathemagical black hole, one that is even less intuitive. I call it the "amazing 15." Start with any positive integer, say 16. Write down all of its divisors including 1 and the number itself. These are 1, 2, 4, 8, and 16. Now write down the digits of these divisors and add up all the digits that are now in the list: 1 + 2 + 4 + 8 + 1 + 6 = 22. Doing the same thing with 22 yields 1 + 2 + 1 + 1 + 2 + 2 which equals 9. But 9 leads to 13, then 5, then 6, then 12 and, finally, 15. After that the 15 keeps on repeating:

1+3+5+15–>1+3+5+1+5=15

Does this happen no matter what number one starts with? By now the choice is clear. Either come up with a theory or write a program to test the process!

The Mailbag

Some confusion has arisen over the swap challenge I posed in *Algorithm* 1.3. In BASIC can you swap the values of variables A and B without using either the built-in SWAP command or another BASIC variable such as C? To emphasize, you may use only the variables A and B in your program. It CAN be done!

Timothy B. Moore of Alexandria, Va. responded to the Collatz conjecture (in *Algorithm* 1.1) with what he hoped was a proof that all hailstone numbers eventually end up in the cycle 4-2-1. He succeeded in showing that if a hailstone sequence ever repeated a value, then it would ultimately fall into this cycle. Unfortunately, he failed to show that a hailstone sequence must ultimately repeat itself. Moore's argument nevertheless adds a valuable insight to our collection of analytical tools.

Among the readers who responded to my hyperbolic string art challenge in *Algorithm* 1.2 were Daniel R. Cohen of Golden, Colo. and Cornelius Groenewoud of Barton, Fla. Not only did Groenewoud achieve the figure in six lines of BASIC, but he wrote the program with no computer to run it on!

In the next Easy Pieces I'll have further adventures and a tough new challenge. Please don't hesitate to write (or send MSDOS disks with ASCII files) if you have questions, solutions or suggestions for future topics. If you wish a reply, please enclose a large self-addressed (#10) envelope and (US residents only) affix sufficient postage.

Dr. Michael W. Ecker
"Easy Pieces" columnist,
ALGORITHM
909 Violet Terrace
Clarks Summit, PA
18411 USA

Until my next column, may your computing be pleasant and may your programming pieces all prove easy!

COMPUTER ADVENTURES

by A.K. Dewdney

Critical events in a computer sandpile

We have seen them in movies and some of us have seen them in real life: Piles of sand that slowly shift with the wind, building and decaying so slowly that we cannot see them move. The desert is a sea in slow motion.

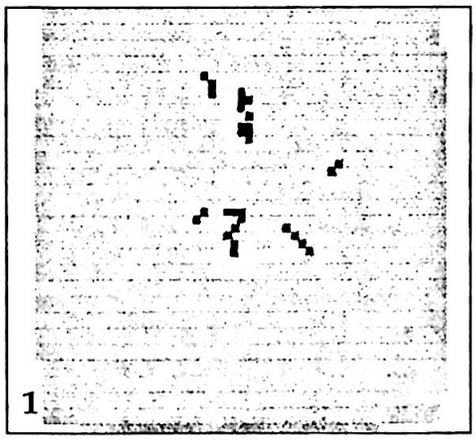
Observed closely, a growing pile of sand shows curious phenomena. There are sudden rushes of sand downhill at all scales of size. Tiny tumblings may occupy only a square centimeter, sheets of sand several meters square may suddenly give way. Landslides. The whole range of phenomena is visible inside an ordinary hourglass.

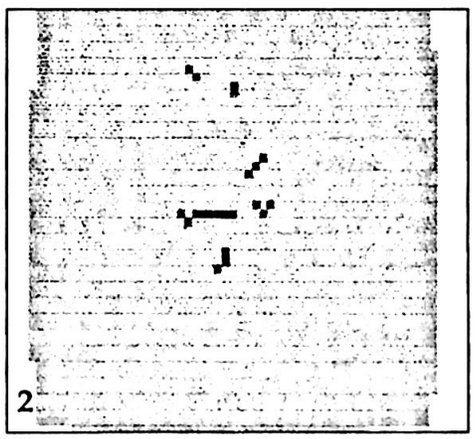
Such phenomena are the bread and butter of scientists such as Per Bak at the Brookhaven National Laboratory in Upton, N.Y. They can be studied, strangely enough, without resorting to sandbox or desert. A personal computer can hold an amazing amount of sand, safely stowed away in the cells of a cellular automaton.

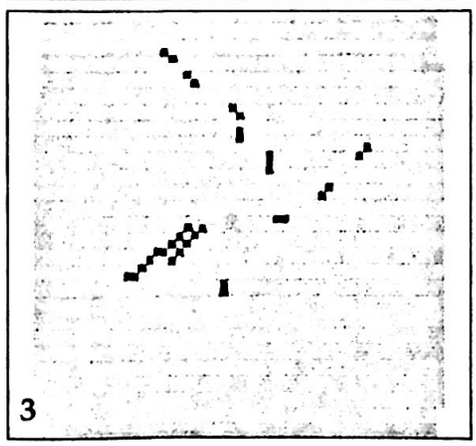
Bak and his colleagues regard sandpiles (real or computed) as but case studies in the larger and more difficult field called nonlinear diffusion dynamics. The aim of the field is to develop theories capable of describing "self-organized criticality." Systems that display this phenomenon are found in biology, economics, physics and other areas. Generally speaking, some systems consist of a great many components that interact in a nearest neighbor fashion. A change in one component can spread to neighboring components and thence further afield. It is remarkable, notes Bak, that such systems so often evolve to a critical state. Ecosystems, for example, are susceptible to noise in the sense that a change in one species can produce changes in dependent species and such changes can even affect still other species. In a sandpile the tumbling of a single grain can set off a chain reaction that spreads all the way down the slope. In sandpiles, ecosystems or any of the other systems studied by Bak, changes may propagate through a small part of the system or through virtually all of it. When the change is over, the system has, in effect, organized itself into a new critical state, ready for further changes.

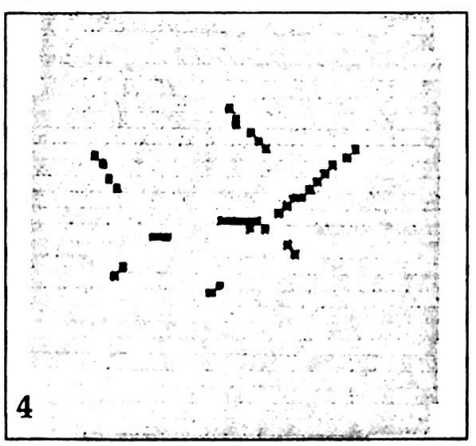
The changes that propagate through a sandpile follow a pattern typical of what physicists call flicker noise. Characterized by the power spectrum 1/f, such "noise" means that there will be a great many changes of short duration in the system (small landslides) and few of longer duration (larger landslides). Flicker noise has been observed in the light from quasars, the flow in rivers, current in resistors and prices in the stock market.

If we build a pile of sand by adding one grain at a time, the pile grows until the slope of its sides reaches a critical value. As more sand is added after this, it slides off the pile. If, on the other hand, we start with a pile that is too steep, it simply collapses to the same critical state. This state is an attractor of sorts for the underlying dynamics. A signpost of the critical state is the size of the avalanches. As the pile builds up there are mostly small avalanches at first. But at the critical state there are avalanches of all sizes, most small to be sure, but some extending down the entire side of the pile. Strangely enough, although there are many different ways to employ local randomness in a cellular sandpile, it makes almost no difference to the appearance of the critical state. Criticality is robust.









TUMBLE
operates on individual grains

```
diff1<-- box (i, j) - box (i-1, j)
if diff1 > 1
        then tumb <-- int (.5 • diff1 • random)
             box (i, j) <-- box (i, j) - tumb
             dbox (i, j) <-- 1
             box (i-1, j) <-- box (i-1, j) + tumb
             dbox (i-1, j) <-- 1
diff2 <-- box (i, j) - box (i+1, j)
if diff2 > 1
etc ...
```

A COMPUTERIZED SANDBOX

The program I call DUNE uses a 100 by 100 array of integers that will represent the height of the sandpile at various points. Specifically, if the array is called *box*, the *i j*th entry, *box (i,j)* will be an integer that represents the height of the pile above the point (*i,j*). A cellular automaton must have rules, of course. The rules I will use here are based largely on what might be called the discrete physics of individual grains. When such rules are applied over the entire array *box*, it is like a wave of computation sweeping over the sandpile. Some-where, here and there, will be slides!

Consider the top few grains in the (*i, j*)th cell. If *box*(*i, j*) is one unit greater than any of the surrounding cells in positions (*i*-1, *j*), (*i*+1, *j*), (*i, j*-1) and (*i, j*+1), then the top grain at (*i, j*) is obviously stable and will not tumble during the current iteration. If, on the other hand, there are micro-pits on any or all sides of the top grain at (*i,j*) it becomes increasingly unstable, the more so the deeper the pits. The instability is captured by a probability that is multiplied by the height difference.

DUNE examines each of the four cells of *box* adjacent to the cell (*i, j*). Take *box* (*i*-1, *j*), for example. The algorithmic fragment of DUNE shown above is called TUMBLE. The variable *diff*1 contains the height diff-erence between the sand in the (*i, j*)th cell and that in the (*i*-1, j)th cell. If the value of *diff1* happens to be greater than 1 then there is some instability. At most half the grains that are exposed to the micro-pit in cell (*i*-1, *j*) could fall into it, since there is no more room in the hole than this. But not even this many grains will fall in. Indeed, a random number between 0 and 1 determines the exact number. If *diff1* is 5, for example, TUMBLE will multiply this number by .5 to obtain 2.5. Next, if the random number happens to be .42, the integer part of the resulting product, 1.05, is 1. In other words, one grain of sand will tumble into the pit at (*i*-1, *j*) in this case.

Once the number of grains to be tumbled has been determined, the actual tumbling amounts to nothing more than an adjustment to the values of *box* (*i, j*) and *box* (*i*-1, *j*). The former is decreased the latter is increased. Following each adjustment to the number of grains in the cells affected, TUMBLE sets the corresponding entries of a special array called *dbox* to 1. This array will figure later on in the display section of the program DUNE.

TUMBLE must handle not only the migration of grains into the cell at (*i*-1, *j*) but also into the cells at (*i*+1, *j*), (*i, j*-1) and (*i, j*+1). But the algorithmic pattern is precisely the same.

The rules for tumbling grains are not exactly short and sweet. As a result, if they are applied at every cell in the array box, it may take some time before a computational cycle is complete (one tick of the cellular clock). For this reason, readers may wish to use a smaller box. Even one that has dimensions 70 by 70 will take less than half the time required by the dimensions I suggested earlier.

Since our simulated sandpile begins with a single grain and since the pile is at first very small, there is no reason for DUNE to compute all 10,000 cells of the array during the initial computational cycles. DUNE therefore uses a variable called *size* that is initially set to 1 and increased thereafter whenever needed. The main algorithm, shown at the bottom of this page begins by setting all the entries of the array *box* to 0 within a double loop (not shown). Then, after giving *size* its initial value, it enters the main loop, in this case, a while-loop; as long as no key is pressed, cycle will follow cycle. As a single grain is added to

DUNE
the main algorithm

```
initialize box to 0
initialize dbox to 0
size <-- 1
while not key
   box (51, 51) <-- box (51, 51) + 1
   for i <-- 51-size to 51+size
      for j <-- 51-size to 51+size
         if size=50
            then for i <-- 1 to 101
                      box (i, 1) <-- box (i, 2)-2
                      box (i, 101) <-- box (i, 100)-2
                 for j <-- 1 to 101
                      box (1, j) <-- box (2, j)-2
                      box (101, j) <-- box (100, j)-2
                 size <-- size-1
         TUMBLE grains at (i, j)
         if flag is set
            then size <-- size+1
   display dbox
```

the top of the pile at the beginning of each cycle, the pile slowly gains altitude. The variable size is changed only when a grain spills off the pile onto the "floor" of the box somewhere.

Within the while-loop the first step is to add a grain of sand to the central cell, the one at (51, 51). The 100 by 100 array is actually 101 by 101. A subarray of radius *size* must be centered on the cell at (51, 51). Therefore the main processing loop runs in both dimensions from *51-size* to 51+*size*. If the value of *size* happens to be 50, DUNE carries out a cleanup process around the outer border of the array. It sets the value of each border cell to two units less than its neighbor toward the center of box. This policy preserves the slope of the pile and prevents sand from piling up around the edge of the box. Readers will notice that once the border is thus adjusted, the variable *size* is set back to 49. In this way DUNE avoids having to deal with cells that have missing neighbors.

Following the cleanup operation, the procedure called TUMBLE is executed. In order that DUNE may know when to increment the variable *size*, a few special instruction must be added to TUMBLE: If a grain of sand is ever deposited in a cell that has one of its coordinates equal to 51-*size* or 51+*size*, a flag must be set so that when DUNE exits from its loop, it may increment *size*. Readers who don't know what a flag is may nevertheless create one. Let it be an integer variable (called *flag*) that starts off as 0 when the double for-loop is first entered. If a grain of sand ever tumbles into the edge of the current array, *flag* is given a value of 1. When the double loop is completed, the test "if flag is set" really means "if *flag* = 1".

Following the change in *size* (if necessary), DUNE displays the current state of the array *box*. Here are opportunities for those who like a graphic challenge. What is the best way to display the sandpile?

Although I could suggest that students of advanced graphics display a perspective view of the pile, this would not necessarily make the most informative display. Wouldn't you most like to see where all the avalanches occurred? That, after all, was one of our purposes in setting out.

Readers who program DUNE may alter the physics of their sandpile by making the factor .5 (in the TUMBLE procedure) into a parameter that is set before a run. With a setting of .1 grains will be "reluctant" to tumble but, when they do, the action is more interesting.

An array called *dbox*, initialized to contain only zeros, records all the tumbling action in the sandpile. Earlier, the discussion of TUMBLE, noted that whenever a value stored in one of the cells of *box* changed, the corresponding entry of *dbox* was set to 1. At the end of the main computational loop DUNE simply displays *dbox*. A mere glance shows where all the action is. For example, the displays at the beginning of this article show clearly several areas of change, some small and some large, in accordance with the 1/f noise law. After being displayed, *dbox* must be reset to 0 for the next round of 10,000 tumblings. Besides *dbox*, DUNE can be improved to display other items such as the height of the current pile or even a profile of the hill taken along a central axis. I will be happy to publish the best sandpile images sent in by readers.

The images for this article were prepared with the able assistance of David Wiseman.

IFS REPRISE

In *Algorithm* 1.4 I described iterated function systems (IFS), brain-child of Michael F. Barnsley at the Georgia Institute of Technology in Atlanta, Georgia. The landscape below was sent in by Bill Mason of Hornitos, California. Mason wrote his program in 1988 after reading an article by Barnsley in *BYTE* magazine. The program, written in Microsoft QuickBASIC, generates the landscape at the resolution of 150 dots per inch.

Henry M. Pollock III of Peabody, Mass. attempted the four fractal elements, earth, air, fire and water. His best image, fire, was based on intersecting circles.

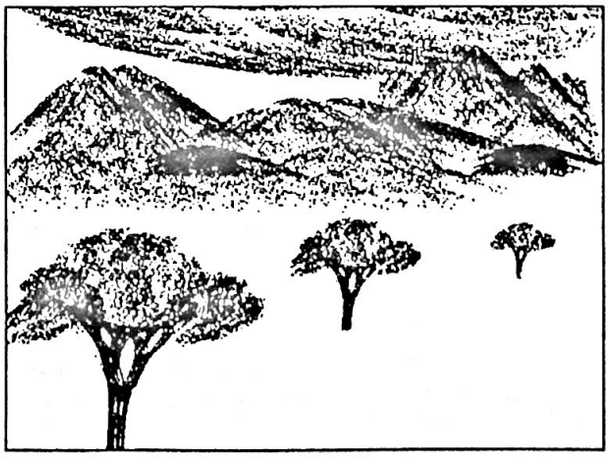
Terry Koyn of St. Louis, Mo. is distributing a program called IFS explorer. It enables a user to define any IFS system he or she likes. Koyn may be reached at 1754 Sprucedale, St. Louis, MO 63146.

I will be happy to receive reader mail at the Algorithm editorial address.

Further Readings:

Malcolm W. Browne,
Scientists view the universe in an ordinary pile of sand,
The New York Times,
Tuesday, May 9, 1989.

Ivars Peterson,
Digging into sand,
Science News, July 15, 1989,
Pages 40 to 42.



The Fantastic World of FRACTALS

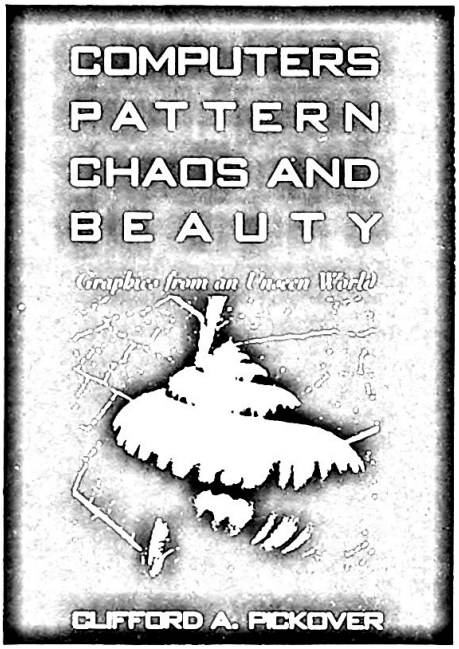
Computers, Pattern, Chaos and Beauty

Graphics from an Unseen World

By Clifford A. Pickover

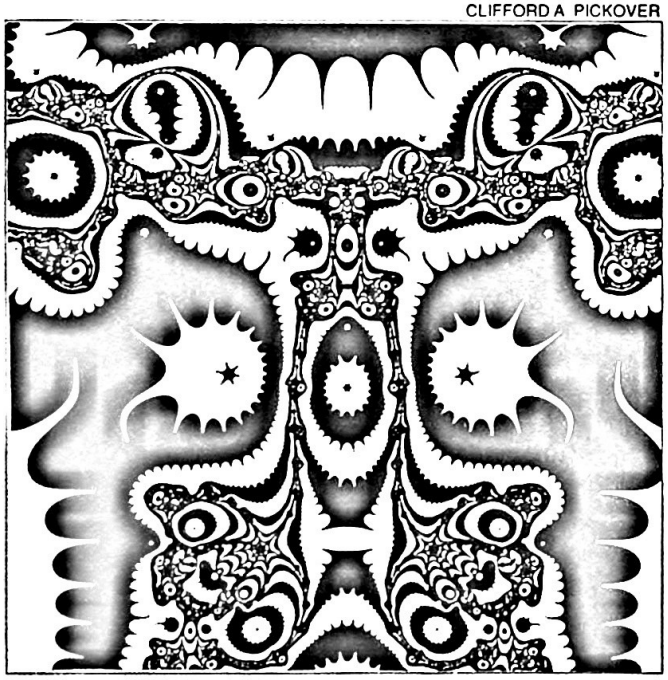
"Chaos and fractals are revolutionary topics these days as they find increasing applications in science, pure mathematics, and computer graphics. Dr. Clifford Pickover, long at the center of this cyclone, has produced a truly stunning survey of its manifold consequences. No human being should pass up the experience of stepping through the portals of this beautiful book into the fantastic new worlds that computers are now exploring in the way a telescope or microscope explores the awesome wonders of nature."

— Martin Gardner, Scientific American



ISBN 0-312-04123-3 $29.95
220 b/w & color illustrations

CLIFFORD A. PICKOVER



"Fascinating. Seems to me to be a completely new way of presenting matters graphically."

— Isaac Asimov

"Van Leeuwenhoek's 20th-century equivalent."

— Omni

Pickover writes in the spirit of Hofstadter's *Gödel, Escher, Bach* and Mandelbrot's *The Fractal Geometry of Nature.* With a combination of text and stunning computer-generated images, Pickover reveals an entirely new method of perception. It is in the field of advanced computer graphics that he finds themes from high-level mathematics and chaos theory to philosophy, art and aesthetics, all interweaving in a unique and beautiful synthesis of science and art.

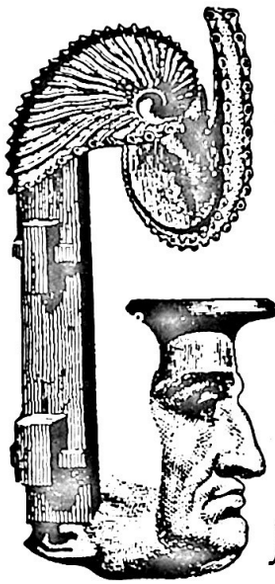
S T. M A R T I N 'S P R E S S

Scholarly and Reference Division ■ 175 Fifth Avenue ■ New York, New York 10010

To order, call 1-800-221-7945 (In NY, (212) 674-5151) or fax (212) 777-6359

PERSONAL PROGRAMS

by Clifford A. Pickover



rowing your own fonts.

Decorative type fonts have been used since the invention of moveable type to catch the eye and ornament the page. With computers it is now possible to store a great many fonts, decorative or not, and to create entirely new ones. My own favorites are "grown" on the spot. Starting from a skeletal framework entered by hand, certain growth processes will produce fonts that appear rustic, weedy, bushy or, in a word, organic.

As a glance at the opening font on this page attests, there is no end to the inventiveness employed in ornamental fonts in the past. In her delightful book, *Bizarre and Ornamental Alphabets* (see Further Readings), Carol Grafton has catalogued all manner of whimsical letters. Some of these sport growths of one kind or another. Why not turn the computer loose on growing its own?

FROM FRAME TO FONT

I will now present the means for doing just this. For readers interested in the science behind the process, the technique employs kinetic growth models in a two-dimensional space. Similar processes model polymerization and aggregation processes that occur far from equilibrium. This means that the user begins the growth process by defining a skeleton of tiny adjacent dots from which growth can occur. The term "growth" simply means that more dots are added next to the original dots in the skeleton. The figure below shows such frames for the letters F and E.

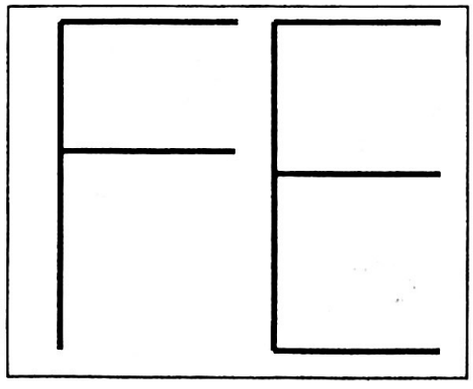
The frame for a grow-your-own font is placed upon a two-dimensional grid, the lines of the frame being strips of adjacent cells in the grid. In the computer the grid will be an array in which letter cells will contain ones (for black). Other cells will contain zeros (for white). To start the growth a black site on the grid is selected by a random number generator. The program then examines an adjacent site in the neighborhood of this one and, if the new site is not black, the program makes it so. Another random site is now selected and the procedure is repeated. The process (called 'Eden growth' or 'reaction-limited monomer cluster growth') is repeated many times.

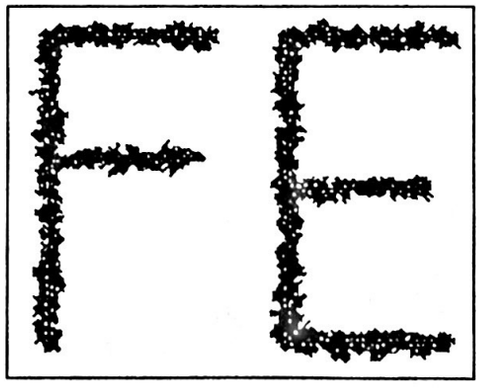
The more iterations the font-growing program allows, the thicker the letter becomes. (see the sequence below.) The user may control the density and structure of the growth by employing more complicated rules. For example, a new site may be disallowed from turning black if it is surrounded by more than a certain number of occupied sites. The last illustration of page 12 shows an interesting but not too useful font obtained by allowing growth only at some of the tips and corners of the letters. Before this, a bushy letter has encouraged growth that is downward and to the left.

Up to this point, the growth I have discussed selects adjacent sites using random +1's and -1's in order to grow to the right, left, up or down in relation to the growth site. In other words, if the growth site has co-ordinates (i, j), the first random +1 or -1 might determine whether the new site will be at (i+1, j) or at (i-1, j) while the second unit would determine a new site at (i, j+1) or at (i, j-1). But rather than use random -1's and +1's to control directions, you can make the present choice of +1 or -1 depend on the previous choice. This deviation from randomness can cause the hairy fonts to favor a certain direction of growth, as in my Personal Programs column in *Algorithm* 1.4. There, I explained how to create plant forms by the use of directed random numbers.

The algorithm I call FONTGRO uses two arrays, *skelx* and *skely*, which hold the *x*- and *y*- data. The array index *i* runs from 1 to the *num*, where *num* is the current number of black cells, the ones where growth has occurred: If *skelx* (*i*)=6 and *skely* (*i*)=5 there will be a black dot at (6, 5). FONTGRO is shown in the box on the next page.

The algorithm assumes that the arrays *skelx* and *skely* already contain the points of the skeletal framework from which the font will





be grown. Consequently, FONTGRO begins with a value for *num* equal to the number of points that the programmer has already used in setting up the frame. I have numbered the statements according to the scheme sometimes employed by Dewdney to make reference to lines easier.

Line 2 specifies 100 generations. In other words, 100 new points will be added to the frame in the course of its elaboration into an organic-looking font. This number may seem too small for some readers. You are, of course, free to change it to suit the size of your letters.

Within the loop, the line numbers begin anew and references to these carry the line number of the loop control statement (2). Thus in lines 2.1 and 2.2 a random number between 0 and 1 is selected, multiplied by *num*, rounded down and incremented by 1. This process yields a random number between 1 and *num*, effectively selecting a point in the current font to grow from.

In line 2.3, *num* is incremented and in line 2.4 two new random numbers, g and h, are selected as the basis for the direction of the new growth. According to whether these numbers are less than .5 or not, the increments *xinc* and *yinc* become +1 and -1 respectively. Finally, in lines 2.7 and 2.8 the increments are added to the point with coordinates *skelx(r)* and *skely(r)* to produce a new point at *skelx(num)* and *skely(num)*.

Note that for the sake of simplicity, I did not have FONTGRO check whether a newly created point happened to sit on top of one already in the growing letter. The net effect is very similar to what the algorithm would generate if it did check, however.

A great deal of fun can be had with such fonts. For example, write a little horror story and invite some children to read a scary tale told by your computer. The title might be

THE THING IN THE ATTIC.

Even as you read, the letters appear to be growing fatter and hairier. With any luck the story won't be needed because the children will be nowhere in sight when your computer comes to it.

FONTGRO

an algorithm for growing ornamental fonts

```
1 num<-- number of points in frame
2 for 100 generations do
  1   r <-- random
  2   r <-- int(r* num) +1
  3   num <-- num +1
  4   g <-- random; h <-- random
  5   if g < .5
            then xinc(num) <-- +1
            else xinc(num) <-- -1
  6   if h < .5
            then yinc(num) <-- +1
            else yinc(num) <-- -1
  7   skelx(num) <-- skelx(r) + xinc
  8   skely(num) <-- skely(r) + yinc
```

THE MAILBAG

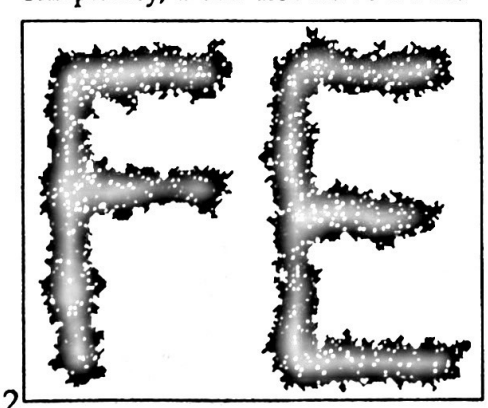
A previous column on cellular automata (Personal Programs, *Algorithm 1.3*) prompted Donovan Smith of El Cerrito, Calif. to modify my design. Smith's new cellular automaton strangely (and accurately) identifies prime numbers. How does it work? The details are a bit complicated but I will be happy to pass the names and addresses of interested readers along to Smith so that he can send them copies of his paper on the subject.

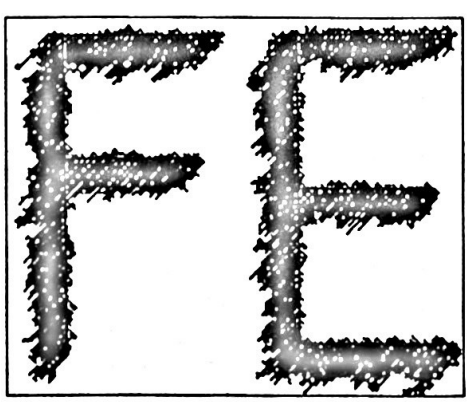
Stephen Ringle of Dover Foxcroft, Maine, has improved the speed of my cellular automaton algorithms. He is also agreeable to sending advice to readers whose names and addresses I send him. Finally, Hal Lichtenstein of San Diego, Calif. was stimulated to explore the inside of the Mandelbrot set after reading Personal Programs in *Algorithm* 1.1. He was astounded to find Mandelbrot-like patterns emerging on his screen when iteration counts corresponding to any prime number were colored black!

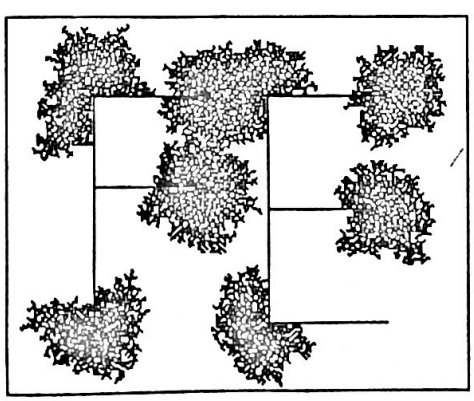
Further Readings:

Carol Grafton,
Bizarre and Ornamental Alphabets,
Dover Books, New York, 1981.

C.A. Pickover,
Markov aggregation on a sticky circle, *Computers in Physics*,
Jul/Aug 3 (4), 1989, 79-80.







ALGOPUZZLE

by Dennis Shasha

One man's poison

It had taken my friend Dr. Ecco a day to resolve the question of the fake coins for the heirs of the Earl of Braycot (see *Algorithm* 1.4). We celebrated that night with Inspector O'Reilly at the Mary Ann, his favorite pub. After a mere two drinks, Ecco fell asleep, perhaps because of his delicate nervous constitution. But the Inspector and I continued to the point of oblivion. My last memory was folk-songs being sung drunkenly (but definitely in key). The next thing I knew, I was being shaken vigorously in my hotel bed in the morning.

"Stop pounding on me," I said, "I feel as if I've been poisoned."

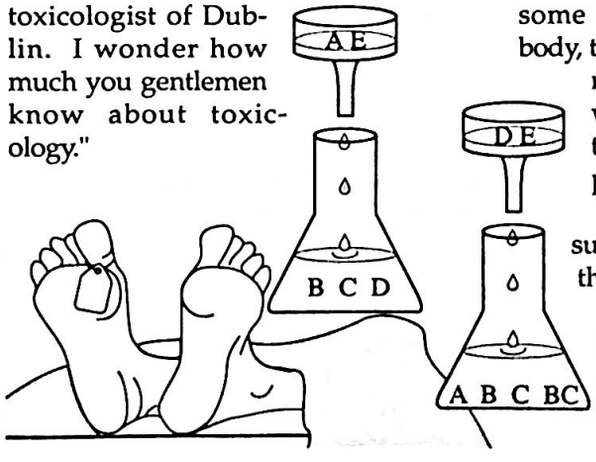
"Wake up, Scarlet," said Ecco, continuing to shake me, "You're not the only one who's been poisoned."

I sat up quickly. There was Inspector O'Reilly with the same twinkle in his eye and pleasant smile on his face. He must have drunk twice as much as I had, but he looked as though he had gone to bed at eight, right after saying his prayers. Next to O'Reilly was a fellow who looked like a cross between a professor and a mortician.

"Our services are needed, Scarlet," said Ecco, "It seems that the Earl may not have died a natural death."

The somber man spoke up.

"My name is Raymond Dooling. I'm the chief forensic toxicologist of Dublin. I wonder how much you gentlemen know about toxicology."



Our blank looks conveyed our response.

"Examining corpses is unpleasant work, most unpleasant work. But necessary, gentlemen, very necessary. It is vital to be able to detect poisons in a corpse, if for no other reason than to bring a murderer to justice. You may recall the alkaloid murders of the early 1800's."

We shook our heads. I, for one, was in no mood for a history lesson, but Ecco was curious, as always.

"As late as 1847 the world's forensic toxicologists faced what seemed an insurmountable challenge in the form of new chemicals called alkaloids derived from exotic plants. At the time the only alkaloids known were morphine, strychnine, quinine and nicotine. No one knew how to isolate these compounds from corpses, so deeply did they penetrate the tissues. It took a Belgian chemist of great talent, Jean Servais Stas, to isolate nicotine by a method that has relevance even to our problem of today.

"Luck played a large role in Stas' discovery. The body of a murder victim, one Gustave Fougnies, had been drenched in vinegar by murderers and later preserved in alcohol by the police. The alcohol-acid mixture penetrated the body and dissolved the nicotine poison. When some fluid was extracted from the body, the alcohol evaporated leaving a residue which, when washed with water, proved to contain the poison. In this case, potash precipitated the nicotine."

"Hmmm," said Ecco, "I'm sure poisons have improved since those days."

"Ah, but so have our methods, Dr. Ecco, so have our methods. The spirit is the same, however. Take the case of the poor Earl. His body shows definite signs of being poisoned but we cannot identify the poison without extensive testing. There are five possible constituents. Once we have identified which subset of them is present in the Earl's body, we shall know the poison and be on our way to collaring the culprit.

"The constituents are arsenic, bismuth nitrate, cadmium chloride, diproionitrile and ethyl arsenate. We'll call them by their first letters, A, B, C, D, and E. We need a testing strategy to discover what subset of these substances was used on the victim. We already know that at least one is present.

"It complicates our inquiry that, besides these five chemicals, two compounds may also be present, BC and ABC. Both are formed spontaneously and rapidly. If there are fewer A than B molecules and fewer B than C molecules, then ABC will form until there are no more A molecules. Then BC will form until there are no more B molecules.

"This is a significant complication," said Ecco.

"It is indeed, I'm afraid. We use membranes of five different types to filter the test fluid. Filter 1 will let B, C, or D pass through but it is impermeable to A, E or any compound. Because the compounds form spontaneously, whatever passes through Filter 1 cannot have both B and C molecules.

"Filter 2 will let A, B, C and BC through but nothing else. Filter 3 is permeable to B alone, Filter 4 is permeable to B, D, and E alone, while Filter 5 will let only A and D through. I trust that is perfectly clear to everyone."

It was not perfectly clear to me, especially in my groggy state. But Ecco nodded enthusiastically.

"Perhaps Dr. Dooling would permit us some practice. Suppose the only possible constituents were A, B and D."

Some readers may wish to pause to consider a strategy of their own.

"To decide what subset of these chemicals were used to poison the Earl in such a simple example, I propose the following: pour the solution through Filter 3. If anything goes through, it is B, is it not? If anything does not go through it is either A, D or both. Redissolve what the filter has trapped and put it through Filter 4. If anything goes through, it must be D. if anything goes not go through it is A. So we can isolate all possible constituents of the poison in this case."

"Exactly," said Dr. Dooling, "The trouble is that the membranes work very slowly. Each takes nearly a week to drain through. Also, our supply of these filters is limited, owing to their extremely high cost. We have one of each kind except Filter 5. Of these we have two. I wonder, Dr. Ecco, whether you can determine within three weeks which constituents are present in our victim."

Here's a chance for readers to match wits with the great Ecco. Devise a series of tests that require only three weeks to perform.

Ecco began working on the problem immediately and soon arrived at a solution. Dr. Dooling was most grateful and apologized that thanks were all he could give. Cities are stingier than rich Earls.

"It's always a pleasure to struggle with fine puzzles, whether they bring material rewards or not," said Ecco graciously. "Should you come upon any more puzzles, I hope you will let me know."

We had no sooner returned to our hotel with O'Reilly and begun to pack our bags than Ecco's wish was granted. Dr. Dooling came to the door and handed the Inspector a note.

After reading it, O'Reilly chuckled. "Eire must be loathe to see you go, my good Dr. Ecco."

"Some of the Braycot heirs are bent on leaving the country," explained Dooling, "We must let them go after a week if we have no real evidence of foul play. But we have some new facts at our disposal. The spectroscopy lab has found that no compound is present in the corpse. In other words, the body contains no ABC and no BC. With the filters at our disposal, do you think we can find the constituents in one week?"

"I'm afraid not," said Ecco instantly. "One week is too little time, even without compounds."

How did Ecco deduce this?

Dr. Dooling looked bitter, at least until Inspector O'Reilly pointed out that article III, section 2.5 of the Irish Criminal Code permitted the police to prevent suspects in suspected homicides from leaving the country for two weeks.

While Ecco pondered the problem of a two-week solution, I suggested to Dooling that if spectroscopy could only determine that just one of the five possible constituents was present, then three filters and one week would suffice.

Which three filters?

At this point, Dr. Dooling was called away and I collapsed back on my bed, whether from mental effort or hangover was hard to say. The next day Ecco's name was all over the newspapers. THE GREAT OMNIHEURIST STRIKES AGAIN, pronounced *The Dubliner.*

Ever the misanthrope, Ecco grumbled about the publicity as we made our way to the airport. Fame, however, does have its advantages. Upon presenting our tickets we were told that we had missed our flight. Luckily an executive with our airline had just read the morning papers and got us a new flight. He changed our tickets to first class and handed them to me.

"There you are, Sir. Quick and sweet, like an ass's gallop."

Solutions to the three italicized challenges will appear next month. Especially clever readers (the ones I call omni-algopuzzlists) may try to discover whether there is a two-week solution to the compound-free problem when there is more than one constituent. If you wish to correspond on these or other puzzles of interest, please use my address as given at the end of this column. In the meantime, here is a partial answer to the algopuzzles posed in the last issue:

Fred Galvin of Lawrence, Kansas deduced that what Ecco announced to the Braycot heirs behind closed doors was that 5 weighings are both sufficient and necessary to determine the counterfeit coins. Galvin's solution (see the box on page 15) is more elegant than my own reconstruction of Ecco's reasoning. He labels the coins A, B, C,... through to J and begins the weighing operations by weighing A, B and C together: "weigh ABC ...".

Aaron C. Brown of New York, NY, found a probabilistic method to find the fake coins in five weighings. According to his probability model, he would be wrong fewer than 1 times out of 100.

I encourage readers to send me their solutions at the following address:

Prof. Dennis Shasha, Algopuzzle
Courant Institute of
Mathematical Sciences
New York University
251 Mercer Street
New York, NY 10012, USA.

Further Reading:

Jurgen Thorwald,
The Century of the Detective,
Harcourt Brace & World, Inc.
New York, 1965.

THE ERROR DEMONS

Not that we avoid our responsibility of error-free publishing, but mistakes will creep in. We think they are caused by demons: typo demons, transcription demons, even printer demons.

In *Algorithm* 1.4 two demons struck, one in the Algofact feature and one at the press.

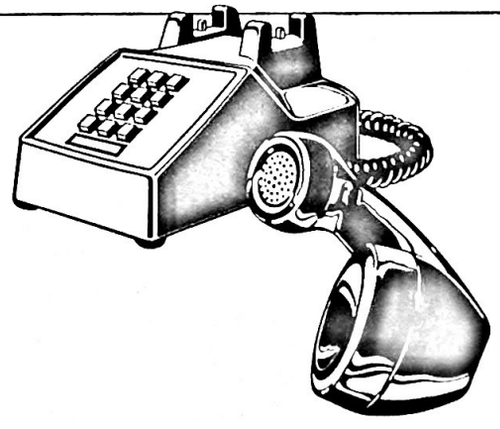
John E. Lauer of Boulder, Colo. and Nick Didkovsky of New York, N.Y. found two errors in step 5 of the neural net algorithm on page 21. The variable *hinputrev* should be *sigma* and *sech* $(h)^2$ should be *sech* (h^2).

The printer demon struck at least 14 times during the printing of *Algorithm* 1.4: a double feed of sheets caused two pages (in this case pages 5 and 24) to come out blank. Automatic collating and binding machines thereafter gave the demon their full cooperation.

Readers who had this problem are requested to notify us so that a good copy can be mailed out free of charge.

Ecco's coin-weighing algorithm

```
Weigh ABC
    if 2 fakes
        then weigh A
             weigh B    / done
    if 1 fake
        then weigh ADE
             if 1 fake
                then weigh AFG
                     if 1 fake    / A is only fake in A-G
                        then weigh HI
                             if 1 fake
                                then weigh H
                                else weigh J    / done
                     if 0 fakes    / one of B, C fake, one of D, E fake
                        then weigh B
                             weigh D    / done
             if 0 fakes    / A, D, E are good, one of B, C fake
                then weigh BFG
                     if 2 fakes    / B must be fake
                        then weigh F    / done
                     if 1 fake    / B is fake or C and one of F, G fake
                        then weigh BHI
                             if 2 fakes    / B and one of H, I fake
                                then weigh H    / done
                             if 1 fake    / B alone is fake; done
                             if 0 fakes    / C
                                then weigh F    / done
                     if 0 fakes    / C is only fake in A - G
                        then weigh HI
                             if 1 fake
                                then weigh H
                                else weigh J    / done
    if 0 fakes
        then Ecco is now in the position of determining which of seven
        coins are fake in just four weighings.  Here is a second chance
        for algopuzzlists to find a solution!
```



Algophone
(519) 432-8042
Fax: (519) 432-8778

ALGOFACT

by Michael W. Lesk

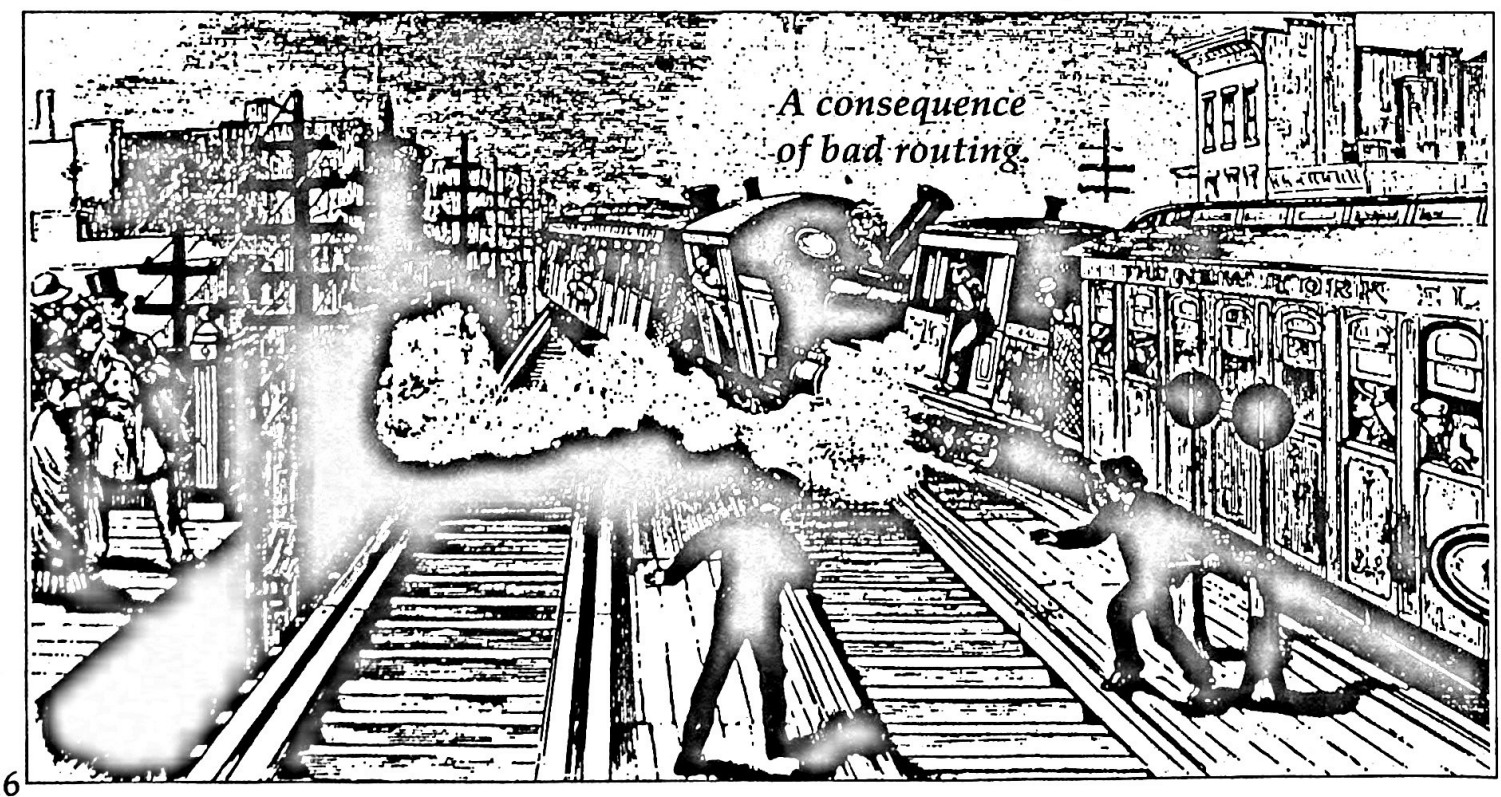
Packets versus circuits: a 19th century case study

Should messages and data be transmitted through computer networks by the method known as packet switching or should a so-called virtual circuit be used? In the first case the routing decision is made locally each time an item is transmitted from one computer to another. In the second case the route has already been determined on a global basis. Which method is best? In the last century, railroads had a similar problem: should trains proceed from signalman to signalman or should they be moved according to complete schedules that are determined in advance? If the analogy really holds, heavily used networks should prefer virtual circuits.

In the 19th century, the Midland Railway in England had a problem. Coal wasn't getting through to London. At this time (1890s) the Midland and all other British railways scheduled their passenger trains but not their slow freights. Railway lines were divided into block sections, each typically a few miles of track., with a signalman's cabin at the boundary between each block section. A coal train would meander from signal cabin to signal cabin, each signalman deciding whether to send it forward or hold it on a siding. He did that by telegraphing the next signalman forward, and asking whether the line was clear. If the next signalman accepted the train, it moved forward one block section, and then the process was repeated. Typically, each signalman was looking at two running lines of track, one in each direction, and a siding or two, although many signalmen found themselves at more complicated junctions.

American readers should understand some of the constraints affecting 19th century British railway operations. Typical British goods wagons (freight cars) did not then have any kind of brake that could be applied while the train was moving. Until the 1960s, British freight trains often had brakes only on the locomotive, tender and guard's van. Even though these trains had to be much shorter than modern U.S. trains, partly because British railways had been built with short sidings and partly because locomotives were much less powerful in the 19th century than they are today, the trains still lacked adequate braking power. Safety was preserved by running at slow speeds; ten miles per hour was common. Even in the U.S., speeds of 18-20 miles per hour were typical for fast freight trains through the first quarter of the 20th century. Meanwhile, passenger trains in Great Britain were already running sixty miles an hour on suitable stretches of track in the 1880s. So a coal train out on the line represented a substantial delay to any passenger train caught behind it.

Time-keeping on passenger trains was important of course. Signalmen were instructed that



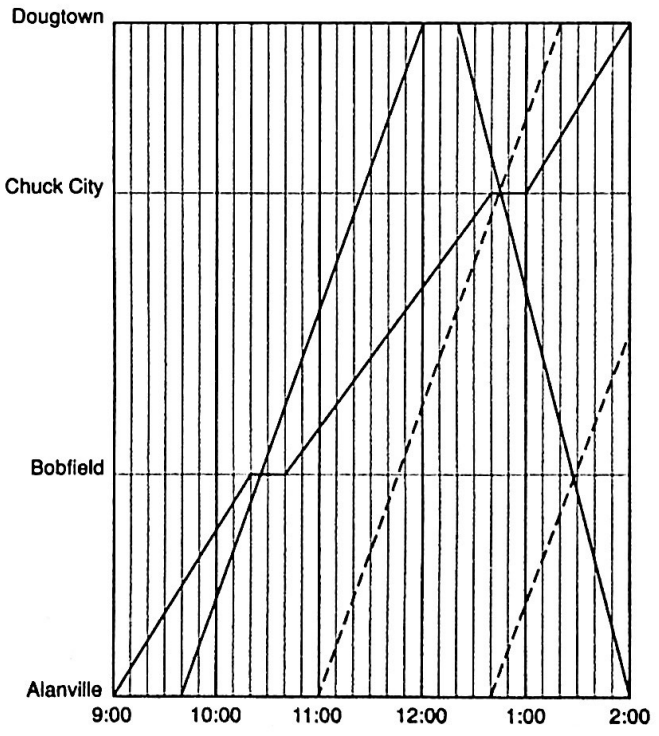
delaying any passenger train was a serious error, while it did not matter much how fast coal moved down the line. So they tended to be conservative and, when in doubt, they put the freight trains into the sidings. But now consider the consequence: suppose you are a signalman, and all your siding space is full of trains and an adjacent signalman proposes to send you another train; you ought to refuse it because if you cannot get one of your sidings empty before it arrives, a main line will be blocked.

You can now imagine the consequences. As the traffic builds up, the sidings fill, and once all the sidings are full nothing can move. The line may be empty but the trains will sit there until the sidings clear from the destination back. The Midland Railway eventually had to pay enginemen for a day's work in which their train never moved, and it had to refuse traffic because it had no place to put the trains. Computer scientists will recognize this behavior as 'thrashing': if too much working memory is needed by the resident processes the CPU spends it time swapping to disk and only rarely does a process get to run. It is also similar to congestion on an Aloha-type network which is so overloaded that most attempts to send a packet collide and fail, requiring retry after retry, without anything actually getting sent. The Aloha network was a predecessor of Ethernet: instead of listening to see if the bus was clear and then trying to transmit, anyone with a packet to send merely transmitted it and listened to see if it had collided with anything.

The normal English solution to railway congestion was to put down additional tracks. Two-track railways became four-track or six-track. The Midland didn't have the money to widen their system. Instead, in 1907 their general manager, Cecil Paget, with his assistant, J.H. Follows, invented the idea of "train paths." Roughly speaking, the idea was that you could accommodate the irregular freight traffic by having a schedule that displayed not only the trains that ran each day, but also showed the slots for additional trains that only ran occasionally. The schedule provided not just for the trains that were, but for all the trains that might be.

In a train path schedule, the designer first works out the maximum capacity of the line. This involves picking minimum intervals between trains, and also deciding on the maximum speed trains can be expected to maintain. Since different classes of trains will run at different speeds, the schedule must present a balance between slow freights, passenger expresses, and everything in between. In addition, both long distance and local services must be provided for. And when trains must meet or pass, the schedule provides for this to happen at a place with a siding.

An example of a train path diagram is shown in the figure on this page. Time reads horizontally across the diagram, and stations are shown vertically. Lines slanting to the right indicate trains in one direction; lines slanting to the left indicate trains in the other direction. The slope of the line represents the speed of the train. Thus, in the example, train 75 runs slowly from Alanville to Dougtown, stopping at Bobfield and Chuck City for twenty minutes. Train 3 is an express from Alanville to Dougtown, which overtakes train 75 while it is stopped in Bobfield. Train 4 is a fast train in the reverse direction, which meets train 75 at Chuck City. Train 4 travels faster than train 3; perhaps there is some downhill running in that direction. Assuming that this is a single track line and that there are no passing sidings except at the stations, the next possible slot for a fast train running from Alanville to Dougtown is an 11:00 departure, which both overtakes train 75 in Chuck City and also crosses train 4 there. This slot is shown as a dashed line on the map. Any departure between 10:00 and 11:00 would be caught behind train 75 and would have to follow it to Chuck City. The next slot after 11:00 for a fast train is at 12:40, also shown as a line, which has a suitable meet with train 4 at Bobfield. Any train leaving around 12:00, say, will have to wait for train 4 at Bobfield anyway. All of these



A train path schedule is like a virtual circuit

possible paths could be filled in, with an appropriate balance kept between the number of fast and slow trains actually needed to work the traffic.

Once the full list of possible paths was made, the job of the train dispatcher became easier. Given a new request for a slow coal train to London, he simply looked down the list for the next appropriate path, and telegraphed all the signalmen that a train would run on that path today. They could now send it on confidently because they knew that it would not interfere with more important traffic. At each point where there was a conflict, the schedule would so indicate and there would be a siding. As the train left the mine, its arrival time in London would be known.

In the United States, a different system of scheduling trains was used. Instead of train paths, which were felt appropriate only for a few lines with very high traffic densities, each train was scheduled by a dispatcher. Most U.S. lines, unlike those in Britain, were single track, and thus the major issue was to arrange 'meets,' the places where trains going in opposite directions would pass each other. The dispatcher would figure out all the meets for a train, and write out a set of orders, something like 'Extra 2553 east, engine 1002, will leave Veederburg at 10:30, and meet extra 402 west at Tipton, extra 214 west at Frankfort, and extra 354 west at Muncie." These orders would be transmitted to each engine crew in writing and the crews would thus know where to expect each opposing train. The orders would also specify which train would hold the main line and which would take the siding at each meeting point, either explicitly or implicitly, by specifying the 'right' of the train. All trains had 'rights' reflecting their relative priority; for example, passenger trains would have the right over freight trains, meaning that in case of delay the freight trains would have to wait for the passenger trains. As in the train path system, the dispatching system involves prescheduling of all trains, and trains would not leave their originating point until it was known when they were expected at the destination.

Safety, of course, depended on a variety of people. The dispatcher must not issue conflicting orders, the signalmen must deliver the appropriate train orders and set their signals in accordance with them, and the engineers must obey both the signals and the orders. If all were not done properly, the result could be as shown in the figure on page16. Collisions are more serious on a railway than on an Ethernet. Surprisingly, despite the many fictional stories about dispatchers writing 'lap orders' (sending two trains headed in opposite directions into the same section of track at the same time), it does not appear that any accident was ever caused by such a mistake in the history of U.S. steam railways.

In computer networks, personal safety is not at the same risk. The physical plant of the network is not subject to bridge collapses, but can be disrupted in other ways. The closest equivalent to a runaway train is probably a worm. Runaway trains would render a line unusable until brought to a stop or derailed. Fortunately, they could not duplicate themselves!

In the beginning years of this century, both British and American railways adopted systems of centralized control for train dispatching. This greatly improved the capacity of the lines to carry traffic. Later on, even more centralization was adopted. The 'Centralized Traffic Control' equipment that became common on main lines in the U.S. let the dispatcher not only write orders, but actually control the signals at long distance, and observe the positions of the trains. Such equipment has made it possible not only to stop increasing the number of tracks on railway lines, but to decrease them: the four-track mainlines of both the old Pennsylvania and New York Central routes have been reduced thanks to the increasing carrying capacity of CTC-track. Thus, proper scheduling of a network can produce real savings in the amount of physical plant needed.

There is even a comparative experiment between dynamic routing and prescheduling. The French railways originally used local decision making; each stationmaster was in charge of his station, and decided which trains moved and which trains waited. When the United States entered World War I, the exhausted French railways were short of staff and equipment after three years of war. Unable to provide the crews to move U.S. supplies to the front, the French railways gave the Americans one railway line from a seaport to the war zone, and the U.S. Army brought its own locomotives, men, and procedures. Thus there was an American railway running in the middle of France, and, watching it in a operation, the French realized that the Americans were getting more trains per day over the line than they had been able to. At the end of the war they converted first the Paris—Orleans line as a test, and later the entire French rail network, to the American system.

Does this sound familiar? The choice is between pre-routing and dynamic routing. Today we have packets or datagrams instead of trains, we have network nodes instead of signalmen, and we have

store-and-forward buffers instead of sidings. In packet or datagram networks, each message is given an address header, and each travels independently to the destination. Each node of the network knows at least one way of moving a message closer to the destination, and each decides as it is looking at the message which way to send it. In virtual circuit networks, before sending any message there is a procedure for opening a circuit, which defines a path from the source to the destination. The routing is decided at the time the circuit is set up and each message sent along the circuit goes the same way, in principle. Some virtual circuit networks are in fact implemented in packets and the network software is required to reassemble the packets into a stream preserving the original order of transmission. The standard analogy to the packet network is the telegraph and the standard analogy to the circuit network is the telephone.

In modern computer networks, we have the same basic problems: In heavy traffic, dynamic routing can overfill buffers and prevent a guarantee of service. Virtual circuits may instead deny a circuit, but once the circuit is allocated the routing is known and the service dependable. Some analysts recommend virtual circuits for heavy traffic, others recommend datagrams in the same situation. (see Further Reading.)

The arguments are familiar to us. Datagrams have the advantage that multiple routings may be used for different portions of the same traffic, perhaps even for different packets in the same message. On the other hand this will require reassembling the message to deliver it in the correct order, a service nearly all customers will require. Datagrams also adapt quickly to changes in the network configuration or load, and minimize complex state memory in the network nodes. Virtual circuits, on the other hand, avoid the need to calculate routings on each packet, and thus reduce the overhead of the message. Pre-assignment of the capacity minimizes the risk that, once having started to send a message, it will be difficult to finish it. In exchange it offers a new kind of error message, namely 'no circuit' (or busy signal). Some users may prefer, in cases of heavy overload, to get half their message through; others may prefer that the system promise them, before starting to accept the message, that it will be able to send the whole message.

One argument that is sometimes made is that preassignment of bandwidth might lead to wasted capacity. In the railway situation, this corresponds to saying that the signalmen can do a better job of fitting trains in, using their knowledge of the immediate situation. For example, going back to the schedule on page 17, if train no. 4 happens to be two hours late one day, then it is possible to fit in a fast train in the other direction leaving Alanville at 11:40 , and arriving at Dougtown at 2:00. However, the evidence from the railway analogy is that this doesn't work; someone with a global view of the schedule can do a better job of fitting in the maximum traffic than a large number of signalmen each able to see only the trains immediately in front of them.

What is the lesson? If we accept the analogy, we should conclude that (a) virtual circuits should be preferred to datagrams in heavy traffic situations; (b) pre-routing is more efficient than dynamic routing. Of course, there are differences between railroad scheduling and data switching. All packets in a data network travel at the same speed, and most data networks do not distinguish different priority classes of traffic. On the other hand, the variety of routings in data networks far exceeds that in railroading, and the ratio of data transmitted to the amount of storage in the network nodes is enormously greater. But don't be too smug about present day systems: after all, no 19th century railwayman could throw away a train of coal and ask the mineowner to please retransmit.

This article is a revised and expanded version of one originally published in the European Unix Users Newsletter in 1987. It appears here with EUUN permission.

Dr. Michael Lesk is the Director of the Computer Science Division at Bellcore, Bell Communications Research, in Morristown, N.J.

Further Reading:

Robert Shaw,
A History of Railroad Accidents, Safety Precautions and Operating Practices,
Northern Press, Potsdam, NY, 1978.

Yutaka Matsushita, Haruaki Yamazaki, and Isamu Yoshida,
"An Evaluation of Virtual Circuits and Lettergram Services,"
Computer Networks,
vol. 3, 1979, pp. 287-294.

Alexandr Butrimenko and Ulrike Sichra,
"Virtual Circuits vs. Datagram - Usage of Communication Resources,"
in *Performance of Computer Systems,*
North Holland, 1979, pp. 525-537.

ALGOFICTION

by A.K. Dewdney

VAT MAN (and Reuben, the Boy Wunder)

From his vat-cave, Vat Man makes nightly forays into the crime-infested streets of Gosham Metropolis, using his incredible powers of deduction to solve crimes that baffle Gosham's finest. How does he do it? Vat man's brain is not in his body. It swims in a high-tech vat deep in the Vat-Cave. Tended by Reuben, the Boy Wunder, the brain has grown to a fantastic 200-pound mass of throbbing grey-matter. It communicates with Vat Man's body via intricate radio equipment inside Vat Man's head. Join us now for the episode called "Vat Man meets Doctor Dejavu."

Vat Man stood musing before the vast tank that contained his brain.

"I'm having that problem again, Reuben."

His voice carried an edge of panic and the pimple-faced sidekick looked up in alarm. The problem had been there from the beginning: was he, Vat Man, inside the vat or outside it?

The episodes always made Reuben uncomfortable. He understood the problem, but only intellectually. He could hardly guess the tortures Vat Man's huge intelligence and sensitivity gave him over the question.

"Here I stand, looking at the very house of my thought, where the sentence, 'Here I stand, looking at the very house of my thought' is being formulated. I declare, Reuben, it gives me vertigo. Am I in there or out here?"

Only at such times did Reuben doubt their enterprise. A sidelong glance at his master did not boost his morale. Long ago Vat Man had been fitted with a prosthetic skull but the delicate radio equipment had overheated. Nowadays Vat Man dispensed with the skull, leaving a profile rather like the false front in a western movie set. Behind an ample forehead, his skull fell abruptly away to reveal a cluster of transformers, batteries and circuit boxes surmounted by an antenna. The arrangement gave him a distinctly idiotic look.

Reuben tried to change the subject.

"I've altered your nutrient, Vat Man. I upped the glucose and cut down on some of the growth hormones."

Vat Man looked at his acned sidekick in alarm. Reuben explained himself.

"If your brain gets any bigger, we'll have to make a new vat. Enough is enough, nichtwahr?"

Vat Man couldn't agree less. The more intelligence he got, the more he wanted. It was simply stupendous to think about a subject -- any subject -- and feel the thoughts tumble out in a veritable Niagara. All at once. A rush of propositions, possibilities, proposals, proofs and refutations. It was wonderful. Take the case of the ...

"Vat Man, Look. It's the Vat-Signal!"

There, on a TV monitor showing the clouds overhead, high above the cave, the familiar Vat-Logo shone eerily in the night: animated bubbles rising from a stylized brain.

They ran to the Vatmobile, a contrivance of Reuben's. It contained a back-up brain in a tank that replaced the car's back seat. The brain had belonged to the infamous Karen Krieg, gun moll and cop killer, the only woman ever to have died in the electric chair. It also happened to be the only brain they could get.

The replacement brain hung peacefully in its tank, unaware that it might be pressed into action at a moment's notice. Perhaps it was dreaming as that vatmobile roared out of the Vat Cave and into the night air.

On their way to Gosham, Vat Man complained to Reuben.

"You may not realize this with your limited powers of deduction, Reuben, but Commissioner Borden doesn't really need a vat-signal. Why doesn't he just telephone?"

"Heilige Zeitgeist, Vat Man, you're right. What a brain!"

The vatmobile sped through the darkening suburban countryside until the towers of the metropolis came into view. It sloshed through the streets of Gosham, pulling up with a screech of rubber tubing before the City Hall. The matter was clearly urgent; Commissioner Borden and the Mayor stood waiting for them on the front steps.

"It's Doctor Dejavu," cried the mayor. "He's on a crime spree."

"Dejavu has carried out his threat," explained Commissioner Borden. "He's perfected the deja-vu machine and now he can paralyze whole blocks of the downtown area, roaming at will to plunder banks and stores. This very afternoon a 10-square block area was blanketed by Dejavu's dastardly device. Thousands of people wandered around, staring off into space and saying, 'I've done this very thing before.'"

"Heilige Lebenswelt," gasped Reuben. "That will give him unlimited power!"

"Unless Vat Man can do something about it," moaned the mayor.

"That should present no particular problem," soothed the mental powerhouse. "The deja-vu field should have no effect on me. After all my brain is back in the Vat Cave, while I am here."

He paused for a moment to look searchingly at Reuben:

"Er, what I mean is, I am back in the Vat Cave while my body is here. Come, Reuben."

Before the anxious eyes of the

mayor and the commissioner, Vat Man and the Boy Wunder entered the vatmobile and bubbled off, heading for the Crime District.

Years ago, the City of Gosham had decided to place an area of abandoned warehouses and crumbling buildings off-limits to all law-abiding citizens. Here the pimps and pushers, gangsters, dope fiends, con-artists, shoplifters, second-story men, bookies, gamblers, hit-men, thieves, fences, murderers, winos, stoolies, rapists, child molesters, psychopaths, bank robbers, and lawyers all hung out. Needless to say, it was a dangerous place to be.

The vatmobile gurgled down Capone Avenue, its tank protruding through the roof. Prostitutes and pickpockets turned to laugh. Some hoodlums threw rocks that narrowly missed the tank.

"Not a pretty place, Reuben," murmured the massive mind.

"Where are we going, Vat Man?"

"Doctor Dejavu needs a power source for his field generator, Reuben. I'm betting he's tapping illegally into one of the power substations. Why don't you get a schematic up on the Vat-Display."

They parked so that Vat Man could inspect the glowing circuit diagram undistracted. Some thieves stole the vatmobile's hubcaps while vandals spray-painted obscene logos on its windows.

"Here, Reuben, here!" Vat Man's finger stabbed the screen. "That's the Richard Stark substation, and look, on the same block there's an abandoned warehouse."

The vatmobile lurched forward, blowing a rat out of its exhaust pipe. As they went deeper into the Crime District, Vat Man recited Dr. Dejavu's history from his encyclopedic memory.

"Born Nathaniel Nerdley in October, 1950, educated at PS 22 and George Plimpton High School in South Gosham, graduated in

physics and psychology from Harvard cum laude, started a PhD at Cal Tech but did not complete it owing to mental instability, operated a pari-mutual network for ten years using the proceeds to finance a private laboratory, holds 54 US patents including the Nerdley Tingle Box, the Hot Flush Coil, and the Proctalgia Projector, entered a severe psychotic phase five years ago, moved to the Crime District and began calling himself Doctor Dejavu. Now we know why."

"Heilige lehrbuch, Vat Man, you know everything!"

"Just routine, Reuben. Ah, this would be the power substation."

Vat Man pulled the vatmobile up to a steel fence with a sign that had been recently altered:

WARNING 500,000 VOLTSuck

"My guess, which is correct with a probability of .86491, is that Dejavu has constructed a tunnel from the warehouse to substation. Let us explore."

The two faded into the murky shadows of the street, edging invisibly along the walls, turning a corner and edging further along past a knot of muggers. One of them spotted Vat Man and pulled a gun.

"Okay, mutha. You lay some bread on this Fred or you gonna get some lead in you head!"

The others laughed until one of them spotted Vat Man's antenna gleaming in the streetlight. "You can't put lead in that head, man. They ain't no head there!"

As the muggers fled screaming into the night, Vat Man and Reuben edged still further along the brick wall of the warehouse and turned still another corner. Here a short flight of cement steps led to a rusty metal door. A stocky man in a grey fedora, carrying an old-fashioned Thompson machine gun, greeted them with little enthusiasm.

"Keep walkin', you weirdos."

"Excuse me," said Vat Man, "aren't you Maggot Malone, small time hood and two-time loser?"

"What if I am?"

"I'm from Thompson Firearms. Isn't that a model '34? What a beauty! I see your magazine clip is broken. Tsk, Tsk. You might lose a round just when you need it most!"

Malone squinted at the machine gun, turning it over under the

naked bulb by the warehouse door.

"Allow me," said Vat Man.

He reached gingerly over to finger the clip. The metal magazine fell into his hand.

"You won't be needing this for a while!"

Before Malone could so much as swing a fist, Reuben had pulled a chloroform gun from his utility belt and shot Malone in the face. Together, Vat Man and the Boy Wunder pulled the unconscious thug into the darkness of the warehouse. Upstairs a light shone behind a partially closed door.

"I'm guessing with probability of .95924 that Nathaniel Nerdley, alias Doctor Dejavu, is in that room."

The two edged silently up the stairs, pausing near the top to listen. There was the sound of glassware tinkling and strange machinery humming.

"Probability .99993," declared Vat Man as he burst into the room with Reuben.

They were greeted by a gaunt, dark man with spectacles and a pointy black beard. His saturnine eyes glinted with recognition.

"Aha, Vat Man!" cried Doctor Dejavu. "I've been expecting you."

"That's enough, Nerdley. You are wanted for questioning in connection with several bank robberies. Reuben and I have come to escort you downtown to Police Headquarters. I trust you'll come peaceably."

"Peaceably's the word, Vat Man, but first I hope you will allow me to demonstrate my new plaything: The Nerdley Dejavu Field Generator. I now have three of them installed and running. Here, on the wall are three switches. The first switch turns on the generator downtown, the second switch activates a smaller generator in this very building..."

As he spoke, Dejavu toyed with a bowl-shaped helmet that was covered with wires and transistors. It was also full of little holes, evidently a modified vegetable strainer.

"This will protect me from the local field while I watch you two wrestle with a present that is already past."

Reuben's eyes glowed with holy hatred.

"You can forget that idea, Doctor Dummkopf. It will have no effect on Vat Man since he is not really here, anyway."

"Actually, Reuben, I must say that I feel myself to be here even though, intellectually speaking, I am perfectly aware that my brain is in the Vat Cave. On the other hand ..."

"Shut up, both of you. Just for that I'm going to turn on the third generator first. This one is far away. Ha Ha Ha."

Doctor Dejavu threw the switch.

Reuben felt nothing. And Dejavu, who had not donned his protective helmet, also felt nothing. But Vat Man fell to the floor writhing in mental anguish.

"Reuben," he cried. "Help me. I was just pondering the eternal problem of my existence when I distinctly recognized that I had already done it. Please understand ..."

His voice broke off and he writhed some more before continuing.

"Please understand that each time I ponder the problem of my true location, there is always a different context. But now everything is the ... same. Not merely the inquiry but an ever-mounting stack of reflections that the inquiry and, indeed, the reflections themselves have already been undertaken. Reuben, how is this possible?"

"Heilige Weltschmerz, Vat Man. It's the dejavu field, but how ...?"

"Simple, my simple-minded friend," crooned Doctor Dejavu. "I've set up another dejavu field generator beside the Vat Cave. Your friend will never recover.

"And now it's time to deal with you, my Boy Wunder. Soon enough, you and that ridiculous radio-controlled meat-wagon will find yourself in my personal vat. Acid-vat, that is. Ha Ha Ha Ha."

"Not so fast you foul fiend," cried Reuben. "I have a switch of my own to flick."

Reuben reached for his utility belt and pressed a switch, cutting in Vat Man's auxiliary brain.

Only when he saw the new expression on his master's face did Reuben regret never having tested the back-up system. And only now, as his master staggered and stared wildly about, did Reuben imagine what it must be like to die in the electric chair only to revive in a strange room.

"Is this hell? I thought the priest was kidding, honest. Who are you guys? Oh, God, that must be the devil."

Karen Krieg, who may as well be called Vat Woman, stared in horror at Doctor Dejavu. For the first time in her life (or death) she sincerely regretted all the killing, especially the policeman. He had been young and handsome, like the youth standing beside her. Was it time to give the devil his due? Not if she could help it.

Dejavu stared, astonished, at the retreating Vat Man. Why was the field no longer working? He moaned in frustration as Vat Man ran from the lab pursued by the Boy Wunder. What had gone wrong? What strange equipment had the Boy Wunder triggered? Dejavu screamed at his henchmen:

"Stop them, kill them!

"Never mind," he muttered, "I'll kill them myself."

Hastily he flicked the local field switch. Unfortunately, he had forgotten to don the converted colander.

"Strange," he muttered, looking confusedly around him, "I could have sworn I just turned on the deja-vu field!"

Meanwhile, Reuben had just caught up with the body of his master in time to say, "Vat Man, er, Ms Krieg. I can explain everything."

At the same moment, Maggot Malone had been prepared to blast the two into eternity with the Thompson machine gun, model '34. But Malone was suddenly confused.

"Hey, you guys! didn't I just kill youse?"

"You already asked us that," said Reuben.

"That's right, too," groaned the thug. "This is weird."

Vat Woman, whose brain lay safely in the Vatmobile over a block away, staggered out into the street while Reuben and Maggot fell ever more deeply into a cycle of false memories.

Vat Woman was becoming rapidly accustomed to hell. It reminded her very strongly of certain parts of Gosham. Maybe she could find a bar, a quiet place where she could have a few and figure out how to avoid the Devil.

She passed two hookers leaning against the wall.

"Hey, baby. Want a date?"

"Screw off, I'm a woman."

The ladies of the night broke into raucous laughter.

"One of those," said one.

"And with a aerial," chuckled the other. "Remind me not to tune into *that* station!"

Vat Woman turned a corner to find herself facing a bar with dingy neon sign in the window:

PARADISE BAR AND GRILL

Inside it was cool and dark, just as she remembered bars in her past life being. Behind the bar, a burly, middle-aged man named Mike wiped his rag absently across the stained mahogany. He was barely in the field, bothered by only occasional fits of bad memory: "Did I wipe this bar already or not?"

"What's your pleasure?"

"Gimme a double anything. I just arrived in hell and I'm not exactly used to it."

"Know what you mean," sighed the bartender. He poured some cheap scotch. "What brings you here?"

"I killed a policeman."

"That'll do it," sighed Mike. He wiped the bar again.

"Wanna know my punishment? They put me in a man's body. I was a woman in real life and now they got me in a man's body."

"We got a few guys like that around," said Mike. Then he leaned confidentially across the bar. "As long as you feel that way, my advice to you is get a boyfriend. This is a very rough neighborhood."

Vat Woman had two, then three more double scotches.

"This is hell all right. This stuff tastes exactly like scotch but I can't feel a thing. Gimme another double!"

"Whoa," said Mike. "That makes five doubles of the finest in the house. You honestly don't feel any effect?"

"I might as well be drinking water," cursed Vat Woman.

With a heart full of remorse, she remembered the handsome youth who had followed her to the warehouse door. He was the only one who seemed to know her. Dare she return to the Devil's lair?

Back at the warehouse, things were much the way Vat Woman had left them. She stared anxiously into the gloom but the devil was still upstairs, gazing at a certain switch. Dejavu had concluded, for reasons that were not yet perfectly clear to him, that the second switch had to be turned off. But every time he reached for it, he hesitated. "I've already tried that!"

Reuben, still pondering with Maggot Malone, whirled when Vat Woman touched his shoulder. Hadn't somebody just done that? And hadn't someone just said those very words?

"C'mon, sweetheart, you and me are getting outa here."

It took a near eternity for Reuben to accompany his master's body down the street. As it happened, they headed in the direction of the Richard Stark substation.

By the time he recognized the vatmobile, Reuben's head was clear once again.

"Karen Krieg?"

"Thank God," sighed the ex gun moll with something close to religious conviction. "What in hell's going on!"

Reuben pointed to the tank inside the vatmobile. He showed Karen her own brain in its nutrient bath.

"But I'm not in there, I'm out here!"

"Feel the back of your head while I pour a little schnapps into your tank."

Vat Woman slumped to the curb, eyes staring dully at the littered gutter. When the schnapps began to circulate in the nutrient bath, she felt a little better. To her it seemed the scotch was finally kicking in.

Reuben had no time to lose. For once, he would be the hero. But first he had to convince Ms Krieg to retrieve Doctor Dejavu from the warehouse. He tried.

"How do I know this isn't really hell anyway and you're just playing games with me, torturing my soul?"

"Trust me. And remember, as he comes out of the field, Dejavu might get violent. But don't worry, we'll have help!"

When Vat Woman had gone off in the direction of the warehouse, Reuben called Commissioner

Borden's office. Within minutes, twenty squad cars were pulled up in front of the electric substation. Criminal activity in the street fell off slightly.

When Vat Woman returned with the evil scientist, there were forty uniformed officers standing just inside the dejavu zone, checking their guns, rechecking them and generally fidgeting. Dejavu was hustled into a squad car, never to trouble Gosham again.

To make a long story short, Vat Woman turned off both fields singlehandedly. This won her the admiration of the Police department, and a promise from the DA's Office not to prosecute her brain for escaping the death penalty. Unfortunately, she had to agree to a temporary retirement to the vatmobile tank so that Vat Man could re-inhabit his accustomed body.

Back in the cave, Vat Man stared moodily at his own enormous brain, suspended moodily in its tank.

"It was awful, Reuben. You have no idea. Imagine a mind as massive as mine trapped in a feedback loop. It was the most painful, not to mention humiliating experience of my life!"

Reuben nudged his master.

"Ah yes, the brain of Ms Krieg. Under the circumstances, I see no reason why we couldn't find a backup body for our backup brain. Incidentally, I also see no reason why Ms Krieg's brain could not be moved in here with, er, me."

"Heilige doppelganger," exalted Reuben, "It'll be just like family."

Further Reading:

Daniel C. Dennet,
Brainstorms: philosophical essays on mind and psychology
MIT Press, Cambridge, Mass. 1986.

BACK ISSUES OF

Algorithm

An increasing number of readers, especially new subscribers, have been inquiring about back issues. What was fun then is fun now!

Algorithm 1.1: The Mandelbrot Set, Hailstone Numbers, Artificial Life
Algorithm 1.2: IFS fractals, Strange Attractors, The Oracle Machine
Algorithm 1.3: Lightning Lines, Cellular Automata, The Volksrobot
Algorithm 1.4: Foxes and Hares, Digital Gardens, Fractal Forms, Neural Nets

To order back issues, send a check or money order along with a copy of the form on the right or a reasonable approximation thereof. We will send you, post-haste, the reading and recreation you have missed.

Mail your order to:
Algorithm
P.O. Box 29237
Westmount Postal Outlet
785 Wonderland Road
London, Ontario, CANADA
N6K 1M6

Please send me the following back issues of Algorithm.

issue	number of copies	cost
1.1	_____	_____
1.2	_____	_____
1.3	_____	_____
1.4	_____	_____

enter $5.00 in U.S. funds for each copy ordered.

I have enclosed total: _____ in payment.

NAME: _____

ADDRESS: _____

_____

_____

MICRO SCOPE

by David Geoffrey

Micro Scope, a new feature in Algorithm, reviews some of the more interesting software, videos, games, books and other "small products" that cross the editorial desk. For more details see the end of this review.

THE RAILRODDERS

Reading the covering letter from Ben Green, Jr. my eye was caught by the following sentence: "I call it 'Frankie and Johnny settle the west while building the Union Pacific.'" That was enough to pique my curiosity. My hand reached for the diskette, popped it into the drive and typed "turmite".

Two points of light, one red and one white, sprang into furious motion on the screen. They started off about two inches apart on the screen each fluttering moth-like and leaving trails behind. Soon, complicated patches of red and white hid their respective movements.

Suddenly, both moths decided to strike out for the edge of their world. But, as they moved, they left railroad tracks behind. Yes. Railroad tracks: ties, rails and all. Their frantic construction brought them all to soon to opposite sides of the screen. Their world was not as large as I had thought.

I was engrossed in trying to figure out how these silly creatures were laying track when one of them ran into the tracks that had already been laid by the other. Without a moment's hesitation, it pivoted smartly into the other's tracks and began to tear up the rails! Rushing thus up the tracks, stealing ties as it went, the vandal moth soon ran into its more diligent colleague. Then things got confusing. Almost too quickly for me to follow, the white moth turned red and the red moth turned white (I think), and then a third moth was born. It was red.

While the first two moths proceeded to tear up all the tracks they had just finished laying, the new moth fluttered around, got its bearings and proceeded to lay track of its own. I settled back for a long watch as the three moths laid track, stole ties, even stole rails, and generally made a wonderful mess of my screen.

Franky and Johnny, for so Green calls his creatures, had not only settled the west (if one progeny counts) but had built and destroyed the Union Pacific railroad many times. Inspired by Greg Turk's turmites (See Further Reading), Green has written a program that runs more than one critter at once.

A turmite is simply a Turing machine that moves on a two-dimensional grid of squares instead of back and forth on a linear tape. A Turing machine has only a finite number of states and, whatever state it happens to be in, it makes decisions characteristic of that state. For example, in one state, if it happens to be on a red square, it may color it black, turn right and move one square. But, in the same state, if it happens to be on a black square it might leave it that color and move one square straight ahead.

In Green's program, the turmites move so quickly, whirling through ever-changing color patterns of their own design, they resemble electronic moths. The program is written in Microsoft C (version 5) and the source code is surprisingly short considering the complexity of the patterns produced. The code is well-written and should be easy to modify by those with other scenarios in mind. In fact Green has suggested several changes that could be made with relative ease. He has assigned a sex to the two initial turmites (The old song goes "Frankie and Johnny were lovers..." but I've forgotten which was the woman.) Green uses sex to determine the outcome of collisions. Several interesting possibilities suggest themselves. Should males fight when they meet? What should a male and a female do? Green has his own answers to these questions but others will have ideas of their own.

Green has promised to supply a copy to anyone who sends him a self-addressed diskette mailer. US residents are asked to supply adequate postage. Non-residents could make a reasonable contribution to mailing cost. Might I suggest that a 5 1/4" floppy be included as well? Each program comes complete with source code. Readers may write Green at the following address:

Ben A. Green, Jr.
2370 Algonquin Road
Schenectady, NY 12309 USA

A GALLERY OF VIDEO ZOOMS

If someone would hurry up and perfect a high definition color, flat screen television, then computers would come into their own as a preferred artistic vehicle. We could view the latest video from Art Matrix as a gallery of slowly changing scenes. Only this technology would do full justice to the humbly titled "Mandelbrot Sets and Julia Sets".

Nevertheless, viewing the video on an ordinary color television is an experience not to be missed. Eager to give full value to those who order the video, Art Matrix have not spared the tape. It takes two full hours to view from beginning to end, pretty long for one sitting. That is why I can see it as the ultimate high-tech art form. Perpetually there, ever changing, evolving, as incredible scene follows incredible scene. I can imagine myself turning to watch as I pass it for the thousandth time. "Ah, yes. I had forgotten how beautiful..."

There are Mandelbrot sets, Julia sets, and bifurcation diagrams. How shall I describe this incredible

production? Let me begin by saying that it was etched, frame by frame, for untold hours of computation on some of the largest computers on Earth. The results are like nothing we can produce on our humble micros in any finite amount of time. The resolution is staggering, the colors stunning and the music track stays with the video in tempo and spirit. Excuse me, I need just a little bit of a Julia fix before continuing...

Ah, that's better. Any reader who has spent any amount of time computing a Mandelbrot or Julia set on a home computer knows how compute-intensive the project is. At any reasonable dwell (the number of iterations of the basic formula needed to produce detail), the "deeper" portions of the Mandelbrot set can take hours for even a big machine to compute. This video shows hundreds of full screen computations as continuous zooms into the set, a cosmic roller-coaster ride into a strange land of complex, unbelievable beauty.

I wish that Art Matrix had provided some kind of manual with the video so that I could understand, rather than merely appreciate, some of the hoops they have put the set through. Here are miniature Mandelbrots crawling out of obscure places in complex space, abstract riots of color where I thought the plane was featureless. And there are contours and valleys inside the set that I had never dreamed possible.

Art Matrix does not zoom in on the Julia sets in their new production.. Instead, they show large numbers of Julia sets in sequence. As the point that defines the Julia sets itself wanders around the boundary of the Mandelbrot set, the Julia sets form a rotating sequence. From the inner borders curled fingers seem to reach out and flick other features across the set, like some Daliesque volleyball match. And yet an inexorable symmetry always pulls the fingers back into themselves.

Art Matrix has found new fractal shapes to explore. Some simply defy description. Others remind me of strange pictures beamed through space from far away planets by some mathematical probe. There are weather patterns on those planets. And there are weird and wonderful land masses that went through mystical seasons as I watched. Did I see new life forms? Who can say?

Those bifurcation diagrams we thought were so simple have hidden their charms well. The vertical line that divides, then redivides and redivides, leads the so-called logistic equation into chaos when the parameter reaches about 3.56. Beyond this the chaos is solid, well, chaos. Except there are what I call "bands of sanity", places where the chaos disappears and only a few lines lead down to the next zone of "madness". A zoom on one of these lines reveals that it is not so simple after all. Here is a complete copy of the original diagram! Ad infinitum. With-in these traceries of webwork there were power lines that tried to suck me into a TRON-like land of compumythical existence. Only my channel changer saved me.

It has to be seen to be believed. For the ear there is an electronic sound track that matches the views, heightens the drama and steers the viewer. It works. If you don't have a large, flat, high definition color monitor already installed in your livingroom wall, throw a fractal party and build an evening around the video. It is currently available only on Earth.

Price: $50.00 (US) plus $3.00 for postage and handling.

ART MATRIX, P.O. Box 880MJ
Ithaca, NY 14851-0880, USA
Phone: (800) PAX-DUTY
Fax: (607) 277-0959

HOW MICRO SCOPE WORKS

This department will review the products and productions that fall under our personal programming mandate. Individuals are favored but companies with the "right stuff" are also welcome. We will scrutinize not only software but videotapes, books and a variety of other media. Readers with programs to share or distribute are asked to follow a few simple guidelines:

1. Document your program as fully as you can. We don't have the time to guess how your program is supposed to run.
2. Send your program on a 5 1/4" or 3 1/2" diskette. We have the facilities to review software on a wide variety of computers. Don't let the fact that you have the only Filbertron Auto-Gleeb 1000 in your city stop you from sending in your software.
3. We will not return diskettes. Nor will we guarantee that a review of your software will appear. We will try to respond to you, however, to let you know our reaction to it.
4. Be sure to include the conditions of distribution with your submission. This can vary from shareware to a full-blown commercial project.

To the extent that these conditions can be interpreted for other media, they apply. Please send your submissions to the Review Editor, care of *Algorithm*.

Further Reading:

A.K. Dewdney,
Turmites and two-dimensional Turing machines make tracks on the plane,
Computer Recreations,
Scientific American, Sept. 1989.
(See also Computer Recreations for March, 1990).

HOW TO READ ALGORITHMS

An algorithm is a generic description of a specific computation. The description does not follow the syntax of any particular programming language but it follows the overall form of a program; it consists of a sequence of informal statements. Even in our algorithms, there is some syntax, however. For example, assignment statements such as *newy* <– *a* - *x* use the left-arrow as a dramatic indication of how assignment statements actually work: the current values of variables that appear on the right of the arrow are combined according to the expression there (in this case, *a* - *x*) and the resulting number is placed in the variable (in this case *newy*). In most computer languages the left-arrow would be replaced by an equal sign (=) or colon-equals (:=).

Another feature of our algorithmic style is the use of indentation to set off blocks of instructions. Such blocks appear inside loops or conditional statements. The following algorithm contains four indented blocks:

```
input corna, cornb
input side
for i <– 1 to 100
    for  j <– 1 to 100
        x <– corna + j x side/100
        y <– cornb + j x side/100
        c <– int (x² + y²)
        if c even
            then plot (i, j)
```

The first block consists of the two input statements and the for-loop. Inside the for-loop is a second for-loop comprising the second block. Inside the second for-loop is a block of four instructions that are only carried out while the inner for-loop is executing. In some programming languages, such blocks are marked off by "begin" and "end" statements. In BASIC, a for-loop is demarked by a "for" at the beginning of the loop and a "next" at the end. The fourth block in this example consists of the plot instruction. This is indented because it is only executed when the condition (in this case, "c even") happens to be true.

In the informal algorithmic style, special commands can be invented and used, provided that they correspond to something a computer can do. For example, "int ($x^2 + y^2$)" means, "take the integer part of the sum of x^2 and y^2." Sometimes the shorthand is extreme: a single, informal line actually corresponds to a whole block of lines. (Readers who want to see something mildly astonishing might enjoy converting the algorithm herein to an actual program).

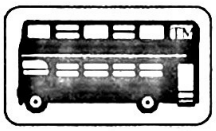
EXPLOR CHAOS

with

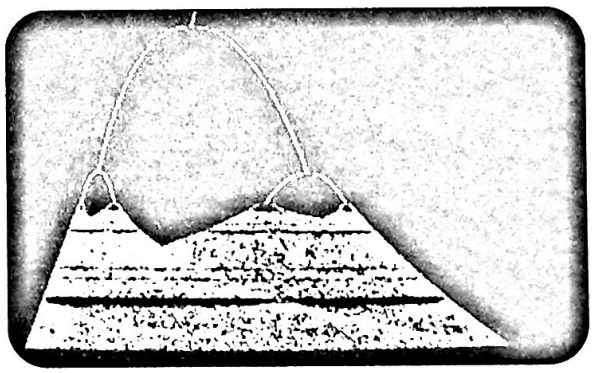
EXPLOR-I

from

TURING OMNIBUS



P.O. Box 1456,
London, Ontario
CANADA N6A 5M2



For PC and compatible computers
with CGA compatible graphics.

EXPLOR-I is a package of programs designed for amateur and professional explorers alike. Select up to eight programs for your own personal package: Julia and Mandelbrot sets, Wallpaper for the Mind, Exploring Chaos, The Invisible Professor, The Cyclic Space Cellular Automaton, Star Clusters Simulation, Computer Popcorn, Biomorphs. In addition, we now offer the Core War Coliseum, the official Core War programming and tournament package. Write for our free catalogue to-day.

BULLETIN BOARD

Software, games and publications from individuals and companies.

New Release
WHAM-DUZLE Graphical Diversion
Experimental Design Kit with source code.
Requires 5.25" PC Compatible with MicroSoft or GWBASIC
Color Monitor optional.
Non-Objective Minimalist Art by rapid tessellation of extended character set.
Send $30 plus tax check or money order:
Christopher Computer, 28 Anderson St.,
Boston, MA 02114-3648

Have you ever *seen* a function of a complex variable? Would you like to really understand differential operators like *grad*, *div*, or *curl*? How about a peek into the 4th dimension?

• • •

Ask about our latest software for PC or Macintosh

Lascaux Graphics 3220 Steuben Ave.
Bronx, NY 10467 USA (212) 654-7429

What is the APL Programming Language?

APL is a quick and efficient tool for exploratory numerical analyses of Algorithm puzzles/articles.

Call the BBS\APL 301-384-3672 3/12/2400 baud, 24 hrs. Many downloadable files (IBM/DOS, not Apple), including 2 free APL interpreters.

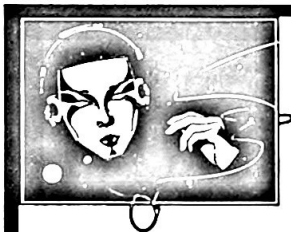
Quelques fiches et menus en français.

Expert Systems on your PC
IBIS by
Intelligence Manufacturing, Co.
Inexpensive, Easy, and Powerful
Educational Version-Shareware $40.00
Development System $250.00

Contact: **D.E.M. Software**
P.O.Box 456 Woodbine NJ
08270 USA (609) 861-5181

AMYGDALA

A Newsletter of fractals & $\mathcal{M}$
Box 219, San Cristobal, NM 87564
Subscription, 6 issues, @ $15.00
Color Slide Supplement,
6 sets, 4 slides each, @ $19.00



J. of B.A.S.R.A.

Journal of the British-American Scientific Research Association Specializing in avant garde science and speculations in science.
A forum for ideas in the sciences.
For information write to: **BASRA**
13 Durwood Place
Madison, NJ 07940

EVERGLADE — **a poem in hypertext by Rod Willmot**

Programmed by its author, a Canadian poet with an international audience, **Everglade** opens the door to new forms of literature. A first in IBM-compatible shareware, PC-SIG disk #1901. To order fully registered send $9.95 and disk size to: **Hyperion SoftWord**
535 Duvernay
Sherbrooke, QC,
CANADA J1L 1Y8

RECREATIONAL & EDUCATIONAL COMPUTING

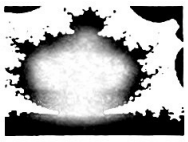
By Recreational Computing Columnist/ Math Professor Dr. Michael Ecker,
with contributed columns by Martin Gardner and others
Send $24.00 for one year of 8 Big Issues, or SASE for more information to:
The REC Newsletter
909 Violet Terrace
Clarks Summit, PA 18411

EduTech's CHAOS Plus...

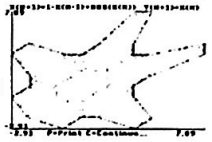
Attractors, Fractals, Automata

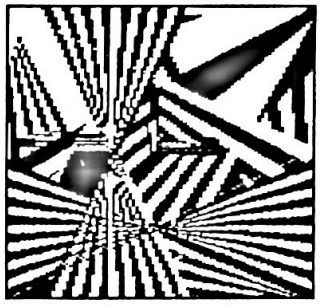
For Apple II Computers (II+,E,C,GS)

$75.^{00}$ US - Money Back Guarantee - Free Catalog



EduTech
1927 Culver Road
Rochester, NY 14609
(716) 482-3151





Hallucinations™ 2.0
Screen Pattern Generator
For IBM & Compatibles
Shareware Disk; $5; Reg. $20.

Write for free catalog

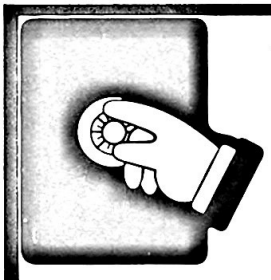
Polymath Systems
P.O. Box 795
Berkeley, CA 94701

FRACTAL RECREATIONS
for IBM PC: VGA, EGA, CGA
Generate your own fractals using **BALLISTIC AGGREGATION TECHNIQUES.** Beautiful fractals resemble coral, bushes, trees and mountains. In color on EGA or VGA. Many hours of fun and education. Menus and Help plus hot keys provide ease of use. Introductory offer: $25.00
Contact: FRACTAL RECREATIONS
21 Wichard Blvd.
Commack, NY 11725-1706

(516) 543-3850

See Inside Music

A videotape of animated musical scores. Music of Scarlatti, Byrd, Bach, Chopin, Webern and others, accompanied by a novel, uniquely effective graphical score that can be understood without any musical training. Form, texture, and melodic contour are revealed in this beautiful and precise representation. VHS, color, 40 minutes. $25 postpaid: Music Animation Machine, 1850 Arch Street #5, Berkeley, CA 94709



A Very Superior and Unique Cryptographic Program for P.C.'s

• The Last Word in Data Security •
39.95 US$ (47.95c)
Cheque or Money Order to
Alex J. Smith,
#1419-3240-66 Ave. S.W.
Calgary, Alberta T3E 6M5

Customized Crypto Processes also available

UNUSUAL SOFTWARE

For your IBM or compatible

*** Metaphysics * Creativity ***
*** Education * Recreation ***

Free catalog, or sample disk $5.00
Rosemary West
P.O. Box 8059
Mission Hills, CA 91346

MegageM
Presents
Fractal Pro

The Premier Fractal Art & Animation System For All Amiga PC's Mandelbrot & Julia Sets and Their 'Cube' Analogs in 256 Colors Magnify up to 10 Trillion - Color Palette Control With Color Cycling
Automatic Sequencing for Superb Smooth Animated Zooms and Pans
Six Directions of Motion: Zoom (In-Out) and Pan (Left-Right-Up-Down)
Handcrafted FAST Assembly Language - Transparent 68020/30 Support
Interactive Full Featured User Interface - Free Tech Support
Only $89.95 + $2.00 Shipping (6.25% Tax in California) - Sample Pics & Catalog on Disk: $2.50
MegageM 1903 Adria Santa Maria, CA 93454 (805) 349-1104

PUZLEPACK - $20.00 US
Solve jumbled words, anagrams, crosswords.
Find all words, any length, in a set of letters.
Fill in "wild card" blanks, fixed or floating.
96,000-word RAM dictionary, amazing speed!
Requires 640K IBM PC compatible or PCjr.

PERMTABL - $15.00
Permutation tables in unique cyclic form. Up to 16 elements, including duplicates. Sets which generate more than 3 pages of cycles are skipped. 130-page printable file plus generating source (BASICA).
Specify 5 1/4" or 3 1/2" floppies. Check or M.O.
RECROOM RECWARE
P.O. Box 307, Pacific Grove, CA 93950

If you have a Commodore 64 or 128

The world of mathematical artistry is within your reach too!
Mandelbrot Set - full control, colour, zoom, high speed.
Cellular Automata - set rule, seed, edge etc.; or auto. Warp speed
Darwin - evolution on-screen in minutes! Scientific Am. May 89
3-D Projection - view math functions & rotate 3-D objects.
Alphametic - high-speed on-screen solution of addition puzzles.
Magic Monkeys - animated, entertaining arithmetic for kids.
$10 each, or $30 for all 6 above:
Also $10 each: **Julia Sets; Affine Transformations**
Ian Adam 4425 West 12th Ave.
Vancouver, BC, Canada V6R 2R3

Algorithm

BULLETIN BOARD ITEMS ARE FREE!

Your first posting on our bulletin board is free. Subsequent postings will cost $20.00 (US) each. *Algorithm* accepts no responsibility for the veracity of claims made about products or services offered herein.

COPY-THIS PAGE

WRITE US A LETTER

Moving? Have a product or service to distribute? Wish a friend to subscribe? Want to suggest a topic? Fill in the appropriate space below and mail the copy to us at:

Algorithm
P.O. Box 29237 Westmount Postal Outlet
785 Wonderland Road South
London, Ontario, Canada N6K 1M6

Name .....

Old address	New address

effective date: .....

If you're moving, let us know by filling out the space on the left. This way you can ensure an uninterrupted flow of fascinating projects and engaging articles.

Perhaps you know someone who would like to subscribe to Algorithm, the Personal Programming Newsletter. If so, copy this page and fill in the space on the right.

PLEASE send me 6 issues of Algorithm for the introductory price of **$29.95**

NAME: _____

STREET: _____

CITY: _____

STATE: _____

COUNTRY: _____ ZIP: _____

PLEASE SEND PAYMENT TO:
Algorithm P.O. Box 29237, Westmount Postal Outlet, 785 Wonderland Rd., London, Ontario, Canada N6K 1M6
(Cheque or money order. No C.O.D.)
Can. Funds $35.95 • Overseas add $4.00 Shipping.

If you have a product or service you'd like to make available to our readers, fill out the space to the left with a rough sketch of your ad and we will set it in type. First appearance: FREE! Subsequent appearances: $20.00/issue.

Is there a topic you'd like to see us cover? Fill out the details in the space below:

Dear Editor: _____

_____

_____

_____

_____

_____

Name: _____ Phone: _____

Address: _____ Date: _____